56
29.

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE INGENIERIA

Análisis Comparativo de Velocidad de Ejecución de Software de Realidad Virtual

T E S I S

Que para obtener el título de

INGENIERO EN COMPUTACION

p r e s e n t a

MAURICIO JACOBO ROMERO

Director de Tesis: Dr. Jesús Savage Carmona

México, D. F. 1998

TESIS CON
FALLA DE ORIGEN

A todas las personas que de alguna forma hicieron posible la realización de este trabajo ...

Análisis comparativo de velocidad de ejecución de software de realidad virtual.

Resumen.

Este trabajo tiene por objetivo analizar el desempeño de tres herramientas de realidad virtual:

- VRML (Cosmo Player versión 1.0.1).
- Open Inventor (su programación en modo ASCII, utilizando el visor ivview, versión 2.1.3) .
- dVISE (dvreality) versión 4.0.1.

Estas aplicaciones se correrán en una estación de trabajo Silicon Graphics modelo O2 y en una supercomputadora Silicon Graphics modelo CRAY-ORIGIN 2000.

Se desea obtener información acerca del tiempo de ejecución, así como de la memoria empleada por estas herramientas. La información recabada se utilizará como parámetro de comparación para observar y analizar cuál software tiene mejor desempeño.

La prueba consistirá en una caminata a través de un espacio virtual que representa al laboratorio de interfaces inteligentes de la facultad de ingeniería.

Índice.

Introducción.

Durante los últimos veinte años, hemos visto como la tecnología ha avanzado a pasos agigantados. En el caso de las computadoras cada 18 meses aproximadamente, se duplica su capacidad (ley de Moore). Sin embargo hemos llegado a un punto en el cual la tecnología de los microprocesadores casi ha alcanzado su límite. Por ejemplo la velocidad de procesamiento. Ante estas limitaciones físicas, se han diseñado algoritmos para trabajar con varios microprocesadores en forma paralela, con lo que la velocidad de procesamiento se puede elevar.

Por otra parte, debido a la ley de Moore el precio del hardware se ha abaratado, concretamente el de la memoria y del almacenamiento en dispositivos físicos (discos duros, cintas,etc). Al tener gran capacidad de almacenamiento, las aplicaciones han empezado a crecer en tamaño. Esto ha acarreado que la mayoría del software sea poco eficiente y que desperdicie gran parte de los recursos de cómputo del equipo en el que está corriendo. Si hace diez años le hubiéramos comentado a alguien que necesitábamos de 16 a 32 Megabytes en RAM y 1.2 Gigabytes en disco duro para correr un sistema operativo para PC (concretamente Windows 95), nos hubiera dicho que estabamos locos y que esos recursos eran demasiado para una PC. Sin embargo hoy en día esa es la configuración básica de cualquier computadora personal. Parece increíble que hace 20 años existían máquinas multiusuarios y multitareas con sólo 640 Kbytes de RAM y que de acuerdo con sus usuarios, las aplicaciones corrían a una buena velocidad. Esto nos muestra como desperdiciamos los recursos de cómputo, hemos perdido el hábito de buscar algoritmos optimizados y de implementarlos de forma sencilla y con un código pequeño.

Lo anterior nos enseña lo importante que son las pruebas de ejecución en un software comercial, ya que como no disponemos del código fuente, no podemos optimizarlo, por lo que lo único que podemos hacer, es buscar el producto comercial que utilice de forma eficiente nuestros equipos. En este caso lo que buscamos es encontrar el producto de realidad virtual que utilice de forma óptima el equipo del Laboratorio de Interfaces Inteligentes de la Facultad de Ingeniería y la supercomputadora CRAY-ORIGIN 2000 de la UNAM.

La realidad virtual ha comenzado a tener un gran desarrollo, debido a la gran cantidad de aplicaciones que puede tener. En el caso del Laboratorio de Interfaces Inteligentes esta tecnología se aplicará al control de maquinaria y/o robots de forma remota. El proyecto consiste en crear virtualmente el lugar en donde va a estar trabajando el robot. El robot "real" podrá ser controlado por un usuario que esté inmerso dentro del espacio virtual, por medio de menús tridimensionales.

Sin embargo antes de comenzar con este proyecto, es necesario saber qué herramienta de software de realidad virtual es la que más se adecúa a las necesidades del proyecto, además de conocer sobre qué plataforma de hardware podemos tener un desempeño óptimo de la aplicación.

Finalmente quiero comentar que tal vez en este momento la realidad virtual parezca un tecnología muy limitada, sin embargo apenas estamos comenzando. Cabe señalar que lo mismo sucedió con Gutemberg cuando inventó su imprenta, ya que aunque su método abarataba los costos de los libros, eran todavía muy caros y poca gente los podía comprar. Por lo que el grueso de la población no soñaba con tener un libro en casa. Sin embargo hoy en día todos tenemos más de un libro en nuestro hogar.

Capítulo 1.

Conceptos Básicos : Medición de Ejecución.

1.1 Medidas de Desempeño.

Una pregunta muy común para los desarrolladores de aplicaciones de software es: ¿Cómo está empleando los recursos de cómputo mi programa?[MAR96]. Para poder contestar esta pregunta es necesario establecer primero cuáles son los parámetros que más nos interesa medir. Para algunas aplicaciones, posiblemente sea más importante la cantidad de memoria empleada que el tiempo de procesamiento que el programa utiliza o viceversa.

Sin embargo muchas veces el programador se encuentra con que el sistema operativo o la plataforma de hardware no proporcionan las herramientas suficientes para poder obtener la información requerida. Debido a ésto, tanto los fabricantes de software como los fabricantes de hardware han tratado de implementar herramientas que puedan solucionar el problema.

En el caso de los fabricantes de software, éstos han implementado comandos que puedan obtener información de los recursos consumidos durante la corrida de un programa, ésto se logra mediante la inserción de código especial en los programas durante su compilación, este código va registrando cada movimiento del programa, sin embargo, ésto puede empeorar el desempeño de la aplicación y por lo tanto dar resultados erróneos[MAR96].

En el caso del hardware, se han implementado contadores de eventos, los cuales registran cierto tipo de acciones o de instrucciones que el microprocesador realiza, proporcionando una contabilidad transparente al usuario, sin necesidad de recompilar los programas. Cabe mencionar que los parámetros que son de más interés para los desarrolladores son: la cantidad de memoria y el tiempo de ejecución empleado por los programas[MAR96].

La ingeniería de software siempre ha considerado a las medidas de desempeño como parte del ciclo de vida de software, incluyéndolas en la etapa de pruebas.

El ciclo de vida de los sistemas de información (software), consiste de una serie de pasos, que nos guían en la implementación y puesta en operación de un sistema de información[1] ; aunque existen varias metodologías y puntos de vista acerca de este ciclo, la mayoría de los autores coinciden en mencionar los siguientes puntos [PRE93]:

1. Especificación o definición de requisitos *(Análisis).* Se establecen y especifican los requerimientos del software.
2. Diseño. El proceso de diseño comienza tan pronto como se especifican los requisitos.
3. Implementación *(Codificación y Depuración).* El diseño se codifica en un lenguaje de programación determinado y sobre alguna plataforma específica .
4. Pruebas. Se comprueba que el sistema realizado cumpla con los requerimientos especificados en la etapa de diseño y que arroje resultados correctos.
5. Funcionamiento y mantenimiento. El sistema se instala y utiliza. Los errores encontrados se deben subsanar.

[1] Éstos pueden ser sistemas de procesamiento de datos, sistemas de información gerencial, sistemas de toma de decisiones y sistemas expertos.

Aunque parezca contradictorio, la etapa de pruebas ha tenido poca atención por parte de los desarrolladores, ésto debido a que en muchas ocasiones sólo se instalaba el sistema (o programa), se verificaba que corriera y que los resultados arrojados por él fueran correctos, sin medir qué tan bien o mal utilizaba el hardware de la máquina. Lo anterior, se debió en gran parte al decremento en el costo del hardware, específicamente de la memoria y de los medios de almacenamiento secundario(discos,cintas,etc.). Al tener a disposición una gran cantidad de recursos de cómputo era más cómodo y barato instalar el software y correrlo. Si su velocidad era lenta, sólo se compraba más memoria.

Por otra parte, las pruebas son consideradas como una función de control de calidad, lo cual no es necesariamente cierto[WIL95]. Existen muchas razones para hacer pruebas y las siguientes son algunas de las más representativas:

- Para checar el correcto funcionamiento del sistema.
- Para detectar fallas.
- Por control de calidad.
- Para satisfacer al usuario.
- Para tener seguridad del buen funcionamiento del sistema.
- Para optimizar el uso del hardware.

Todas las justificaciones anteriores son válidas. Sin embargo sólo del 10% al 15 % de los desarrolladores están entrenados en las técnicas que permiten llevar a cabo una adecuada aplicación de las mismas[1].

En un proceso maduro de pruebas usualmente se comienza haciendo pruebas para verificar que el sistema está funcionando bien y se culmina al llegar a realizar pruebas para detectar fallas y para optimizar el uso del hardware. Un buen esquema de pruebas debe de ser parte integral del desarrollo de un sistema [WIL95].

Diseñar pruebas puede llegar a ser, en algunos casos, un reto más grande que el desarrollo del sistema en sí. Ésto, debido a que las pruebas deben tener una calidad y confiabilidad más alta que la del software que se está evaluando. Pruebas que sean de mala calidad, sólo servirán para probar software del mismo tipo [WIL95].

La etapa de pruebas permite a los desarrolladores e ingenieros de calidad medir el éxito o fracaso de un sistema con respecto a sus requerimientos. La calidad es definida muchas veces como una medida que el consumidor establece de acuerdo a el número de problemas(requerimientos) que el sistema le resolvió y a como éste emplea los recursos de computo con los que cuenta el usuario. Revisar sistemas sin tomar en cuenta las expectativas del usuario, es jugar con la integridad de la aplicación [WIL95].

Para poder medir el comportamiento de un programa, es necesario que éste arroje resultados correctos, ya que en caso de que nuestras pruebas lo alteren podremos detectarlo con facilidad al checar la información que nos regresa.

---

[1] Gilb,Tom and Graham, Doroty.(1993) *Software Inspection: An effective Method for software Project Management.* Addison-Wesley, Inc.

Las siguientes definiciones nos ayudaran a aclarar las ideas anteriores:

- Un error es una acción humana que produce resultados incorrectos. Por ejemplo el escribir una línea errónea de código.
- Una falta es un paso lógico incorrecto, o una incongruente definición de un dato en un programa. Una falta es comúnmente referida como defecto o "bug".
- Una falla es la inhabilidad de un sistema o programa para ejecutar sus funciones de acuerdo con los requerimientos especificados.

La evolución de un error a una falla puede ser considerada como la principal causa de los retrasos en los tiempos de entrega de los sistemas, además de dar baja calidad a los productos finales. Razón por la cual no causa sorpresa saber que la mayoría de las compañíias gastan aproximadamente un 50% de su tiempo total en la reparación de defectos[WIL95]. Los errores generalmente provienen de :

1. Requerimientos obtenidos de forma errónea, contradictorios o incompletos.
2. Por otra parte, si los requerimientos se adquirieron de forma adecuada y no presentan ningún problema, el número de pruebas que hay que realizar es muy grande y por consiguiente es muy probable que si no se diseñan de una forma adecuada, arrojen resultados falsos, lo cual nos llevaría a tratar de corregir fallas inexistentes.

Muchos de los errores pueden reducirse(pero no eliminarse por completo) a través de un desarrollo y ejecución planeada por medio de una tecnología automatizada de pruebas. Esta tecnología nos da cinco pasos a seguir. Cada paso debe ejecutarse en paralelo o adelantándose al desarrollo del sistema para ser exitoso:

1. Planeación.
2. Desarrollo.
3. Tecnología con Desarrollo.
4. Ejecución.
5. Análisis.

Comenzar con un plan sólido de desarrollo de pruebas es tan importante como comenzar con el desarrollo de un sistema, es decir, en esta etapa tenemos que fijar metas y objetivos a cubrir, procurando que sean claros, a corto plazo y no contradictorios.

Incrementar la satisfacción del usuario y reducir el trabajo innecesario son las metas del desarrollo de pruebas. Estrategias de pruebas efectivas y eficientes, así como métodos que proporcionen más productividad, harán que los ingenieros se enfoquen en el desarrollo de los sistemas, en lugar del mantenimiento y soporte.

En la etapa de Desarrollo y Tecnología con Desarrollo, se diseñan las pruebas que se le van a aplicar al sistema. Éstas pueden ser:

- Pruebas funcionales. Ignoran el código fuente de los programas y se enfocan únicamente en las salidas o resultados generados como respuesta de las entradas que se le proporcionaron al hacer las pruebas.
- Pruebas estructurales. Toman en consideración el código fuente y otros elementos de configuración de los productos o aplicaciones bajo prueba. Este tipo de análisis requiere del código fuente o en su defecto de una detallada especificación del diseño para analizar los detalles de la implementación.
- Pruebas de confiabilidad. Son hechas para determinar si las funciones requeridas se han ejecutado sin falla, bajo condiciones(requerimientos) específicas durante un determinado período de tiempo.
- Pruebas de regresión. Verifican que las modificaciones hechas al producto no causen efectos inesperados en él y que éste siga cumpliendo con los requerimientos especificados.

Las pruebas funcionales no toman en consideración como es que está implementado el sistema, es decir, su código fuente, ya que lo ven como una caja negra. Sin embargo, las pruebas estructurales, de confiabilidad, y de regresión si toman en cuenta el código fuente.

Por otra parte, el modelo "V" es otro método que ha sido propuesto para hacer pruebas a lo largo del ciclo de vida del software. Éste se muestra en la figura 1.1 [WIL95].

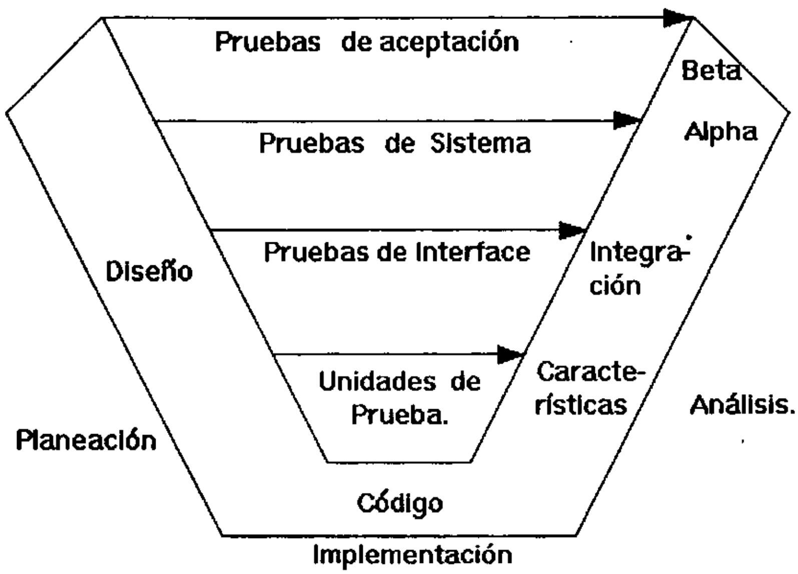


Figura 1.1. Modelo de pruebas "V"

Como podemos observar en la figura 1.1, si nos movemos hacia abajo en la "V" acercándonos al código, se van necesitando más detalles acerca de la construcción e implementación del sistema. Otro punto importante de resaltar, es el hecho de que hacia la punta derecha de la figura 1.1 se menciona las versiones preliminares del sistema, las cuales nos proporcionan un tipo de pruebas y condiciones de las mismas que muchas veces se omiten o pasan desapercibidas para el grupo de desarrollo [WIL95].

Finalmente en las etapas de ejecución y análisis, se llevan a cabo las pruebas y se dan a conocer los resultados, los cuales nos ayudarán a corregir o mejorar detalles del sistema.

Hasta aquí hemos hablado de los métodos que existen para hacer pruebas a los sistemas de información, sin embargo hace falta mencionar cuáles son las medidas que éstos toman en cuenta para realizar su labor. Debemos recordar que las pruebas no se hacen únicamente para detectar fallas, sino también para observar el comportamiento del sistema en diferentes tipos de plataformas de hardware. Las medidas mas comunes son:

- Tiempo de respuesta (Turnaround). Tiempo que tarda un programa en retornar resultados desde que se mandó a ejecutar en la máquina.

- Cantidad de MIPS (Millones de Instrucciones Por Segundo) y MEGAFLOPS (Millones de Instrucciones de Punto Flotante por Segundo) que realiza el sistema en un intervalo de tiempo determinado. Esta cantidad es importante debido a que el tiempo de procesamiento en algunos equipos es bastante caro, por lo cual requerimos de tener una medida aproximada de qué cantidad de operaciones son las que realiza nuestro software, y en base a ésto decidir qué es más conveniente para nosotros, ésto es, si gastamos en un equipo de cómputo o rentamos los servicios de terceros.

- Espacio en memoria y en almacenamiento secundario. Éste es uno de los factores que se deben de estar checando constantemente, debido a que estos recursos son los más demandados por los usuarios y un uso incorrecto de éstos podría afectar el trabajo de todas las personas que son usuarias del equipo.

- Troughput .- Es la cantidad de trabajo que una computadora puede hacer en un período de tiempo dado. Históricamente el troughput ha sido medida comparativa para equipos grandes de computo, una unidad clásica de esta medida es el número de trabajos que puede completar en un día.

- Stride .- Es el espacio en memoria existente entre elementos consecutivos de un vector.

Este tipo de factores nos dan una base para poder emitir un juicio del comportamiento de nuestro sistema. Como podemos observar, en la mayoría de ellos está involucrado el tiempo como principal parámetro [DAN97].

Como ya se menciono, las pruebas son consideradas como una actividad aburrida y poco creativa, sin embargo es necesario cambiar esta actitud, ya que como hemos visto es una labor dinámica y gratificante, debido a que siempre estaremos buscando que tanto nuestro hardware como software esté trabajando de forma óptima, por lo que ésto siempre será un gran reto [WIL95].

1.2 Estrategias para obtener el tiempo de ejecución del Software.

En la actualidad podemos observar cómo las revistas especializadas en computación nos dan su punto de vista acerca de diferentes productos de software, tomando en cuenta principalmente la cantidad de memoria empleada por el programa o paquete y la cantidad de Megabytes en disco duro que se emplea en su almacenamiento, así como la velocidad de respuesta del mismo. En este tipo de pruebas lo que se persigue es encontrar el producto comercial que sea más óptimo para ejecutar una determinada función, tomando en cuenta los parámetros anteriormente mencionados.

Sin embargo estas pruebas muchas veces se hacen en equipos de cómputo cuyas características favorecen a un determinado producto; sin embargo bajo condiciones típicas de uso, estos resultados pueden variar, por lo que las pruebas más confiables son aquellas que se llevan a cabo por organizaciones no lucrativas tales como universidades, asociaciones de consumidores u organizaciones que no dependan de manera directa o indirecta de alguna compañía de software cuyos productos participen en la prueba.

En la actualidad existen dos grandes plataformas de software y hardware en las que se pueden desarrollar sistemas. La plataforma PC dominada por Microsft (DOS y/o Windows en cualquiera de sus versiones) y la plataforma UNIX, en la cual podemos encontrar a una gran variedad de fabricantes y de equipos, que van desde estaciones de trabajo hasta supercomputadoras.

Dentro de la plataforma DOS y/o Windows, el tiempo de ejecución se mide por medio de cronómetros, ya que el sistema operativo no proporciona ningún tipo de herramienta que nos de este dato. Una de las posibles razones por la cual no se tenga implementado un comando para esta función, es debido a que la estructura interna del sistema operativo no permite realizar una operación de contabilidad como ésta, por otra parte los microprocesadores Intel de la familia 80X86, no tienen ninguna función de este tipo soportada por hardware [MAR96].

Dentro del sistema operativo UNIX, nosotros tenemos varios comandos que nos ayudan a llevar una contabilidad tanto del tiempo de ejecución de un programa, como del número de MIPS y/o MEGAFLOPS que éste haya ejecutado. Ésto debido a que bajo esta plataforma, la contabilidad es una de las funciones más relevantes del sistema operativo. UNIX al ser un sistema multitareas necesita de este tipo de información para hacer una administración eficiente de los recursos de cómputo con los que dispone el equipo. Aunado a ésto, los fabricantes de hardware han implementado funciones en los microprocesadores que facilitan esta contabilidad. Como información adicional cabe mencionar que este tipo de microprocesadores son de tecnología RISC y sus velocidades se encuentran alrededor de los 200 Mhz [MAR96].

El sistema operativo de nuestro interés es IRIX versión 6.x[1], que es la versión de UNIX de Silicon Graphics. Este sistema en su versión 6.4 tiene actualmente la supercomputadora CRAY-ORIGIN 2000, mientras que las estaciones Silicon Graphics en su modelo O2 tienen IRIX 6.3[2].

IRIX nos proporciona una serie de herramientas(comandos) para medir el desempeño de un programa o sistema de información en una forma sencilla. Los comandos más utilizados son el *timex(1)*[3] y *ssrun(1)*, los cuales se apoyan en funciones implemantadas en los microprocesadores de la compañía MIPS[4] [DAN97].

Antes de continuar, vale la pena hacer un pequeño paréntesis para recordar cómo trabaja el sistema operativo UNIX. Este sistema operativo se caracteriza por ser multiusuario, multitarea y por poder correr en máquinas con múltiples procesadores. Para lograr ésto, se basa en un esquema de trabajo el cual consiste en dividir los programas en procesos, éstos se pueden ejecutar de forma simultánea en uno o varios procesadores. Un proceso es una abstracción de un programa en ejecución, es decir, es un programa corriendo en el CPU de la computadora, éste incluye el contenido de la memoria, registros del CPU, contador de programa y banderas de estado. A cada proceso se le asigna un intervalo de tiempo de ejecución en CPU llamado quantum [TAN93].

La vida de un proceso se muestra en la figura 1.2.

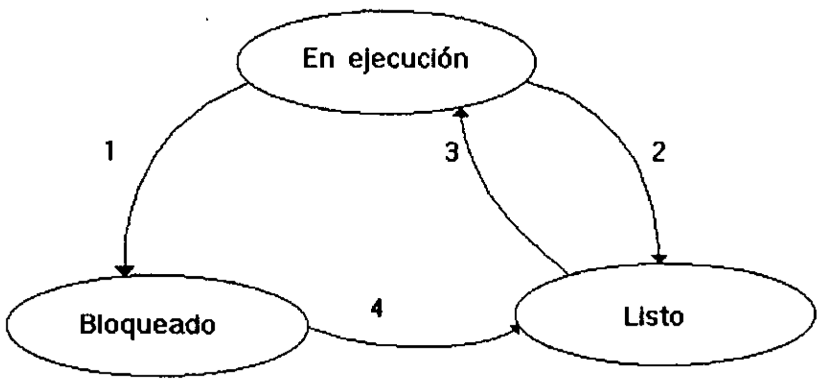


Figura 1.2. Ciclo de vida de un proceso en el sistema operativo UNIX.

Debido a que UNIX es multiusuarios y multitareas debe llevar un control acerca de qué proceso se tiene que ejecutar y cuanto tiempo debe éste permanecer en el CPU. Ésto se lleva a cabo mediante un despachador de procesos, el cual mantiene a los procesos en los tres estados ilustrados en la figura 1.2 . Las transiciones mostradas en esa misma figura se dan de la siguiente manera:

1. El proceso se bloquea "duerme" en espera de datos o de algún recurso.
2. Termina el tiempo asignado en CPU para el proceso.
3. El proceso tiene otra vez tiempo de CPU asignado.
4. Los datos o recursos requeridos por el proceso se encuentran disponibles, por lo que ya se encuentra listo para ejecutarse [TAN93].

Una vez recordados estos conceptos revisemos la primera herramienta de medición que nos da IRIX. El comando *timex* nos proporciona solamente tiempos de ejecución. *Timex* regresa tres medidas:

---

[1] La "x" hace referencia alas versiones que van desde la 6.0 hasta la 6.4 .

[2] Información tomada de la página de Silicon Graphics. *http://www.sgi.com*

[3] El número entre paréntesis indica la sección del manual en línea en el cual se encuentra la información de este comando.

[4] Los modelos de microprocesadores utilizados actualmente por Silicon Graphics son: R5000,R8000 y R10000.

- Tiempo real, es el tiempo transcurrido desde que se mandó ejecutar el programa hasta que éste terminó.

- Tiempo de usuario, es el tiempo empleado para correr el código del usuario.

- Tiempo de sistema, es el tiempo gastado por las llamadas al sistema hechas por el programa.

Frecuentemente el tiempo real es más grande que la suma del tiempo de usuario y el tiempo de sistema, ésto debido a que los procesos pueden haber gastado tiempo "durmiendo", por ejemplo, si éste estuvo esperando por un procesador desocupado antes de poder correr o tal vez estaba esperando algún dispositivo de entrada/salida(I/O). Todos los tiempos regresados por *timex* están en segundos [DAN97].

Una salida típica del comando sería la siguiente:

```
berenice32 10% timex  prueba1
 "Hola mundo"

real                    0.03
user                    0.01
sys                     0.01

berenice32 11%
```

Este comando tiene una serie de opciones, que son descritas en el manual en línea que proporciona UNIX[1].

El comando *ssrun* por su parte nos da información un poco más detallada acerca de la corrida de un programa, ya que nos proporciona además del tiempo total de CPU gastado, estadísticas acerca de cual fue la función de nuestro código que más tiempo gastó. Ésto es muy útil cuando estamos tratando de depurar un programa, ya que por medio de esta estadística podemos ver qué rutina es la que necesita ser optimizada [DAN97].

Para conseguir estas medidas, *ssrun* checa cada determinado tiempo la dirección a la que apunta el Program Counter(PC) del CPU y verifica a qué función de nuestro código pertenece, a partir de estas muestras es que hace su estadística [DAN97].

La salida que da *ssrun*, viene en un archivo llamado:

```
nombre_del_programa.pcsamp.<pid>
```

donde "pcsamp" es el método empleado para obtener la estadística que arroja el comando y "pid" es el identificador de proceso del programa.

Esta información está en formato binario, por lo que para poder verla es necesario utilizar el comando *prof(1)*[2]. A continuación se muestra un ejemplo de ésto:

---

[1] Esta ayuda se obtiene por medio del comando *man de UNIX*. La sintaxis es:

man nombre_del_comando

[2] *"prof"* está descrito en la sección del manual en línea.

```
berenice32 17% ssrun  -pcsamp pi
berenice32 18% prof pi.pcsamp.2345


-------------------------------------------------------------------------
Profile listing  generated Tue Nov 4 13:55:38 1997
    with:        prof pi.pcsamp.2345
-------------------------------------------------------------------------

samples  time     CPU    FPU     Clock      N-cpu  S-interval   Countsize
    353  3.5s  R10000 R10010  195.0 Mhz.    32      10.0ms       2(bytes)

Each sample cover 4 bytes for  every 10.0ms  ( 0.09% of 10.6800s)

-------------------------------------------------------------------------

-p[rocedures]  using pc-sampling.
Sorted in descending order by number of sample in  each procedure.
Unexecuted procedures are excluded.

-------------------------------------------------------------------------


samples      time(%)         cum time (%)           procedure(file)
   1068       11s(100.0)     11s (100.0)          approx_pi   (pi:pi.c)
   1068       11s(100.0)     11s (100.0)                TOTAL
```

La columna de time(%) indica el porcentaje de tiempo gastado en esta rutina , en la columna de procedure podemos ver el nombre de la rutina que gasto el tiempo listado en la columna anteriormente mencionada.

Sin embargo *ssrun* tiene algunas limitaciones; no soporta corridas largas, debido a que los contadores de tiempo están limitados y no son confiables cuando se mide el desempeño de un programa que emplee mucho tiempo para llevar a cabo su función. Además de que para que *ssrun* corra es necesario tener las librerías que fueron incluidas durante la compilación del programa, ya que de otra forma no nos dará ningún dato acerca de la corrida [DAN97]

Ssrun tiene varias opciones que pueden cambiar el algoritmo de muestreo del programa, sin embargo la opción *pcsamp* es la mas general de todas, las demás opciones son descritas en el manual en línea que proporciona UNIX y en general todas se basan en el muestro de un registro de los empleados por el programa durante su corrida, por ejemplo el Stack Pointer.

1.3 Importancia del tiempo de ejecución.

El tiempo de procesamiento es uno de los parámetros más cuidados por todas las empresas e instituciones, ésto debido a que su costo es muy elevado. En el caso de las supercomputadoras, la hora de procesamiento llega a tener un costo desde 1000 hasta 10000[1] dólares.

[1] Información recabada de la lista supercomputers Top 500. *http://www.netlib.org/benchmark/top500.html* .

En la U.N.A.M. por ejemplo, en el departamento de supercómputo de D.G.S.C.A. (Dirección General de Servicios de Cómputo Académico) existe un área encargada de optimizar los códigos de los investigadores, ésto con el fin de poder utilizar de una forma adecuada el tiempo de procesamiento en cualquiera de las dos supercomputadoras.

El costo de la hora de procesamiento se obtiene en base a los costos de mantenimiento del equipo, así como al valor que éste tuvo al momento de su compra.

Un problema al que se enfrentan los sistemas interactivos, es el tiempo que se pierde en la entrada de datos, los cuales son obtenidos de algún dispositivo periférico, generalmente del teclado o del Mouse. A diferencia de éstos, los sistemas de procesamientos de datos son los que mejor aprovechan los recursos de la máquina, ya que se encuentran limitados solamente por la velocidad de acceso al disco duro de la máquina [DAN97].

Cuando se está programando cualquier tipo de aplicación, se recomienda hacer código pequeño y emplear algoritmos optimizados. Si seguimos la recomendación anterior, migrar un sistema a otras plataformas será sencillo y no nos causará graves problemas.

En el caso de los productos comerciales, lo que se busca es que éstos aprovechen al máximo los recursos del equipo. Por ejemplo un programa que fue optimizado al máximo en una máquina de un sólo procesador, es posible que corra más lento en una máquina de múltiples procesadores, ésto debido a que el programa no fue adecuado para correr en una máquina paralela, por lo que no aprovecha todas las ventajas que este equipo ofrece, sino que muy posiblemente se encuentre con que este tipo de recursos obstaculicen su labor. Cuando se quiere migrar un programa a otra plataforma, es necesario conocer todas las características del nuevo equipo, ésto con el fin de obtener un buen desempeño en nuestra aplicación. Volviendo a los productos comerciales, debemos de tener cuidado en las especificaciones del paquete, ya que si el fabricante no nos proporciona información acerca de en que equipo y bajo que características corre su software, muy probablemente no podamos utilizarlo de forma óptima.

Las aplicaciones en tiempo real son las que más requieren de recursos de cómputo; pero también pueden ser las que más los desperdicien, ésto debido a la interacción con los usuarios. Áreas como la realidad virtual, la visualización de datos, el reconocimiento de patrones y el procesamiento digital de imágenes son algunos ejemplos de este tipo de aplicaciones.

El problema principal al que nos enfrentamos en la actualidad es que debido al bajo costo del equipo y a la disponibilidad de recursos (tanto económicos como de cómputo), muy poca gente (entre la cual podemos encontrar desde usuarios finales hasta desarrolladores) optimiza sus códigos, razón por la cual las aplicaciones consumen una gran cantidad de tiempo de procesamiento. Debemos tener en cuenta que no sólo el tiempo de procesamiento en una supercomputadora o en un mainframe es caro, sino que cualquier recurso de computo que se utilice tiene un costo asociado, por lo que se le debe sacar el mayor provecho posible. Para lograr ésto debemos programar buenos algoritmos y tratar de reducir el código. Por otra parte cuando se compre algún paquete se software se deben analizar a conciencia sus características para poder aprovecharlo al máximo y no tener un desempeño pobre.

Existen muchas herramientas para medir el tiempo de ejecución del software. Algunas han sido hechas por universidades y otras por compañías interesadas en esta área. Sin embargo cada una de ellas está orientada hacia fines muy específicos, por lo que su utilización dependerá de las necesidades de cada usuario[1].

[1] Para mayor detalle de este tema en :

"UNIX Test Tools and Benchmarks" de Rodney C. Wilson. Ed. Prentice -Hall .

Finalmente cabe mencionar que siempre es recomendable aprovechar al máximo cualquier equipo de cómputo que se tenga al alcance, para ello nos podemos auxiliar de muchas herramientas(tanto de software como de hardware). sólo es necesario elegir la que más se adecúe a nuestras necesidades. Es importante señalar que toda esta teoría ha empezado a cobrar una mayor fuerza, debido a que las limitaciones técnicas en el diseño de los microprocesadores ya no permiten obtener velocidades más grandes, por lo que es necesario buscar caminos alternativos para elevar la velocidad de las aplicaciones, es por eso que han cobrado fuerza los conceptos de trabajo distribuido y trabajo paralelo, así como todo lo referente a la optimización de algoritmos. Esta tendencia es claramente visible , ya que fabricantes de microprocesadores como Intel, IBM, Digital (DEC) y Hewlett Packard han empezado a implementar contadores de eventos[1] en hardware para poder ayudar a los diseñadores de software en la optimización de sus códigos [MAR96]. Por lo que tal vez durante los próximos años escuchemos hablar cada vez más acerca de la importancia de hacer algoritmos y programas eficientes.

[1] Los eventos son aciones o instrucciones que lleva a cabo el microprocesador.

Capítulo 2.

Conceptos Básicos : Graficación por computadora y Realidad Virtual.

2.1 Graficación por computadora : Orígenes.

Se podría pensar que la teoría que sustenta a las gráficas por computadoras fue hecha en su totalidad durante la segunda mitad del siglo XX, sin embargo ésto no es así. El primero en tratar de hacer estudios serios acerca de como se podría representar de una manera más fiel el espacio que nos rodea, fue el pintor e ingeniero Leonardo Da Vinci, en sus estudios de perspectiva.

A mediados de la década de los 1940's, un ingeniero de nombre Jay Forresters fue elegido para encabezar un grupo de investigadores del Instituto Tecnológico de Massachusetts (MIT). Este grupo era responsable de diseñar y crear el equipo necesario para el entrenamiento de pilotos y para efectuar pruebas aerodinámicas de los nuevos diseños de aviones. Por la magnitud del proyecto se llegó a la conclusión de que una computadora digital podría ser una excelente herramienta para este proyecto. Debemos recordar que la primera computadora digital, la "ENIAC" ya había sido construida y aun no existían computadoras comerciales [MOR95].

Se comenzó a construir una nueva computadora digital, la cual era conocida como "Whirlwind". Después de algunos años, el objetivo del proyecto cambió de un simulador/probador a un sistema de radar computarizado para la defensa norteamericana. Al igual que los radares normales, el "Whirlwind" podía desplegar el mapa de un área geográfica en una pantalla (CTR) y podía graficar representaciones de aviones. El 20 de abril de 1951 fue la demostración de este sistema. El Whirlwind fue el primer proyecto de graficación por computadora [MOR95].

Por ese tiempo un productor de cine llamado John Whitney, estaba interesado en componer música y gráficas por medio de computadoras. En 1958 empezó a experimentar con computadoras analógicas para crear arte. Whitney usó una computadora M-5 antibombarderos y creó animación visual; para controlar los movimientos de la cámara empleó computadoras analógicas. Este aparato fue utilizado para crear efectos especiales en el cine durante la década de los 1960's [MOR95].

En 1959 se crea el primer sistema de dibujo por computadora, DAC-1 (Desing Augmented by Computer), que fue desarrollado por General Motors e IBM. Este sistema permitía al usuario introducir una descripción 3-D de un automóvil así como manipularla, es decir, se podía rotar y tener diferentes vistas del modelo. Sin embargo es hasta 1964 cuando se libera en la Conferencia de Computación en Detroit [MOR95].

El paso más significativo para las gráficas por computadora, fue la creación del programa de dibujo llamado Sketchpad. Este sistema fue creado por Ivan Shutherland (en el MIT) en 1961. El Scketchpad permitía al usuario utilizar un lápiz óptico para dibujar simples cuadrículas en la pantalla de la computadora, guardarlas en tambores y/o cintas magnéticas y recuperarlas después. Otro elemento importante del Sketchpad fue que permitía modelar objetos. Cabe mencionar que los primeros objetos fueron dibujados en monitores vectoriales y no con tecnología raster. En ese mismo año Steve Rusell (también del MIT) creó el primer video juego "Spacewar" [MOR95].

A mediados de la década de los 1960's las compañías TRW, Lockheed-Georgia, General Electric y Sperry Rand, comenzaron a producir equipo de cómputo que trabajaba con gráficos. Por otra parte, IBM introdujo su sistema gráfico: el IBM 2250, la primera terminal gráfica disponible comercialmente. En 1963 Doug Englebart del Stanford Research Institute inventa el primer "Mouse"[1]. Aparecen también en este año las tabletas digitalizadoras [MOR95].

Se publica el algoritmo de eliminación de superficies ocultas en modelos tridimensionales; sin embargo el principal objetivo de los investigadores seguía siendo incrementar el realismo de las imágenes generadas por computadora. Durante los últimos años de esta década Warnock y W.J. Bouknight trabajaron en el sombreado de objetos tridimensionales con color. Ésto marcó el nacimiento de la tecnología raster, la cual consiste en representar las imágenes y los modelos de los objetos mediante una serie de puntos en la pantalla, cada punto es llamado "pixel" [MOR95].

Uno de los primeros métodos utilizados para crear sombras estuvo basado en los trabajos de un físico y astrónomo del siglo dieciséis llamado Johann Lambert. La ley de Cosenos de Lambert nos dice con qué intensidad es reflejada la luz por un objeto cuando ésta lo golpea. Ésto permitió agregar color a los objetos sólidos creados mediante modelos alambrados[2]. Este método fue mejorado tiempo después y es comúnmente llamado "Flat Shadding". En este método cada polígono del objeto tiene el mismo color, variando en tonalidad de acuerdo a la posición de la fuente de iluminación. Ésto provoca que la calidad de la imagen sea dependiente de la cantidad de polígonos que tiene el modelo, mientras más tenga más real se verá el objeto [MOR95].

En la década de los 1970's se introducen los gráficos por computadora a la televisión. La compañía Computer Image Corporation (CIC) comienza a construir complejos sistemas de hardware y software tales como ANIMAC, SCANIMATE y CAESAR [MOR95].

El método de sombreado "Flat Shading" podía hacer que un objeto pareciera sólido, sin embargo los polígonos se notaban con facilidad, por lo que las escenas se veían poco reales. Aunque se podían agregar mas polígonos a los modelos para mejorar la calidad de las escenas, ésto incrementaba la complejidad de los mismos, con lo que se tornaba lenta la ejecución del render[3]. Para resolver este problema, en 1971 Henri Gouraud presentó un método para crear la apariencia de una superficie curveada por medio de interpolaciones de color a través de polígonos. Este método de sombreado de objetos 3-D es conocido como "Gouraud Shading". Uno de los aspectos más impresionantes del método de Gouraud es que requiere del mismo número de operaciones para sombrear una escena que el método de "Flat Shading", ésto provocó un crecimiento espectacular en la calidad de las imágenes generadas por computadora [MOR95].

En 1973 ya se empleaban técnicas como la llamada pixelización para crear mosaicos computarizados. En 1974 Ed Catmull publica su tesis doctoral. Su tesis cubría el mapeo de texturas, el *z*-buffer y el rendering de superficies curvas [MOR95].

[1] Mouse.- Dispositivo de entrada que facilita el manejo de interfaces en modo caracter y en modo gráfico.

[2] Un modelo alambrado es la representación de un objeto mediante una cuadrícula o malla.

[3] Render se denomina al programa que calcula el color de los pixeles en la escena de acuerdo a las fuentes de luz que afectan a éste, así como de acuerdo a la profundidad del objeto en la escena.

El mapeo de texturas le dió a las gráficas por computadora un nuevo nivel de realismo. Este método consiste en tomar una representación 2-D de la superficie de un objeto y aplicarla sobre un modelo 3-D; de la misma forma en que se pone papel tapiz en una pared. El *z*-buffer auxilia en el proceso de remover superficies ocultas, utilizando "celdas". Las celdas son similares a los pixels, solo que almacenan la luminancia de un punto específico de la imagen, además de su profundidad. Una coordenada *z* representa la profundidad, mientras *x* y *y* representan la posición horizontal y vertical respectivamente. El *z*-buffer es un área de memoria dedicada a mantener la información de la profundidad de cada punto en la imagen. Hoy en día muchas estaciones de trabajo de alta calidad gráfica tienen implementado en hardware el *z*-buffer [MOR95].

Poco tiempo después Maldelbrot publica un libro titulado "The fractal Geometry of Nature". Este libro mostraba como sus principios de fractales podían aplicarse a los modelos hechos por computadora de fenómenos naturales tales como las montañas, las costas y los bosques [MOR95].

Mientras tanto la compañía Xerox había decidido enfocarse a las gráficas por computadora en blanco y negro. Por otra parte Alvy Ray Smith crea el primer programa de dibujo que utiliza paletas de colores de 24 bits. Investigadores del NYIT (New York Institute of Technology) desarrollaron scanners para lápices ópticos. Estos dispositivos crean el alambrado de un objeto mediante el empleo del lápiz óptico y modelos de yeso a escala del objeto [MOR95].

En 1976 la "Association of Computer Machinery (ACM)" organiza por primera vez la conferencia anual de gráficos por computadora SIGGRAPH. Diez compañías exhibieron sus productos ese año. Para 1993, esta conferencia reunió a 275 compañías en más de 30000 exhibidores [MOR95].

En diciembre de 1977 una revista llamada Computer Graphics World aparecía por primera vez. Hoy en día continua siendo la revista de mayor vanguardia en el tema [MOR95].

Durante los últimos años de la década de los 1970's Don Grenberg de la Universidad de Cornell fundó un laboratorio de graficación por computadora, cuyo objetivo era producir nuevos métodos de representación de superficies de una forma más realista. Su colega Rob Cook crea modelos de luz que se aproximaran lo más posible a los reflejos plásticos. Cook buscaba crear un nuevo modelo de luz que le permitiera simular objetos con brillo metálico. Este nuevo modelo consideraba la energía de la fuente de luz así como las intensidades y los brillos [MOR95].

La década de los 1970's cerró un capítulo muy importante para las gráficas por computadora debido a la increíble cantidad de aportaciones que se hicieron en esos años.

Ya hacia el comienzo de la década de los 1980's SIGGRAPH era el evento de mayor importancia para el mundo de la computación gráfica. La compañía de cine Lucasfilm incluye efectos especiales por computadora en la película *Star Trek: The Wrath of Kahn.* Años después la película *Wilow* ponía por primera vez en pantalla el efecto de "morphing". El morphing consiste en la transformación de un objeto a otro. Ésto se logra mediante un tratamiento especial de imágenes [MOR95].

Nace en esta década la técnica conocida como Motion Caption, que consiste en almacenar el movimiento humano mediante sensores colocados en el cuerpo de un actor que se mueve de acuerdo con una trayectoria predefinida, ésto con el fin de ahorrar el pesado trabajo de diseñar modelos matemáticos para animaciones que requieran de una gran exactitud en sus movimientos. Las películas *Tron* y *Flash Gordon* impulsaron el desarrollo de esta técnica [MOR95].

También en 1988 la compañía DeGraf/Wahrman Productions presenta en SIGGRAPH 1988 a "Mike la cabeza parlante". Mike era una animación 3-D interactiva. Usando controles especiales la animación podía interactuar con los conferencistas participantes [MOR95].

En mayo de 1990 Microsoft libera Windows 3.0. Una interfaz gráfica de usuario similar a la de las computadoras Macintosh de Apple, dando pauta al desarrollo de las interfaces gráficas (GUI's)[1]. Años después se crea el estándar de ventanas Xlib y sus sucesores Athena y Motif[2]. En este mismo año la compañía NewTek saca al mercado la tarjeta "Video Toaster" para la computadora personal Amiga de Comodore. Esta tarjeta tenía capacidades de animación 3-D y soporte de paleta de colores de 24 bits, con software incluido, además de efectos de video digital y generación de caracteres. AutoDesk pone en el mercado su primer producto de gráficos 3-D, "3D Studio" versión 1.0 [MOR95].

En 1991 Disney y Pixar anuncian un acuerdo para crear en un lapso de dos años la primera película completamente hecha con animación por computadora. Sin embargo la película fue estrenada hasta 1995 [MOR95].

En Octubre de 1992 es liberada la primera versión de *httpd(1)*[3], el programa para UNIX que hace posible el intercambio de documentos de hipertexto, usando como medio de transmisión la red de computadoras Internet. Éste es el nacimiento del World Wide Web (WWW). Aparecen los visualizadores de hipertexto[4], Mosaic 1.0 de la NCSA (National Center for Supercomputing Applications) y Netscape 1.0 de Netscape Comunications Inc. [RUD97].

En 1993 Wavefront adquiere Thompson Digital Image (TDI) [MOR95]. La película *Jurasic Park* hace uso de la animación por computadora para la creación de dinosaurios, los cuales fueron modelados en Alias Animator y animados con software de Softimage. La piel de los dinosaurios fue puesta con el software de Pixar "Renderman". Un año después en marzo de 1994 esta película ganaba un Oscar por efectos especiales [WER94].

En 1994, Mark Pesce y Brian Behlendfor imaginaron la posibilidad de construir una interfaz virtual, tridimensional, con multimedia animada e interactiva, compartida por varios usuarios simultáneos para lo que entonces era una gran novedad: el Web. No podía lograrse todo ello mediante interfaces propietarias. Para ser útil debía de ser abierta y gratis. Se requería todo tipo de personas para desarrollar un estándar y el Internet era el lugar adecuado para encontrarlos. Así fue como nació VRML (Virtual Reality Modeling Language). De ser a principios de 1995 un pequeño grupo de personas participando en una lista de discusión, era ya para SIGGRAPH 95 (en agosto) un estándar apoyado por compañías como Silicon Graphics e IBM [RUD97].

En 1995 Silicon Graphics compra a la compañía de supercomputadoras CRAY Research, con lo que se coloca a la cabeza de las compañías productoras de hardware para gráficos. Aparecen nuevos modeladores, siendo el más renombrado Alias|Wavefront, el cual permitía el trabajar con NURB's[5] en lugar de polígonos. Aunado a

[1] Graphic User Interface, consisten en dar al usuario un entorno de trabajo basado en el empleo de ventanas gráficas, donde el manejo de comandos es de forma casi intuitiva, facilitando el trabajo del usuario final.

[2] Estos proyectos consisten en una serie de bibliotecas gráficas que permiten programar ambientes de ventanas de una manera más sencilla y de forma portable.

[3] El número entre paréntesis indica la página del manual en línea de UNIX en el cual se encuentra su descripción completa.

[4] El hipertexto consiste en poner video, audio, imágenes y texto en un mismo documento.

[5] Las curvas NURB's, son curvas generadas por medio de interpolación entre diferentes puntos llamados "puntos de control", y dan una mayor suavidad a los modelos sin necesidad de introducir una gran cantidad de polígonos y por consiguiente de información.

ésto poseía herramientas de render que proporcionaban una excelente calidad en la imágenes que se modelaban en él. Aparece también el microprocesador Pentium MMX, el cual emplea sus capacidades principalmente para multimedia[1].

Durante 1997 aparecen muchos aditamentos(plug-ins) para los navegadores de Internet (browsers) que permiten la visualización y navegación de espacios virtuales hechos con VRML. Por otra parte Pixar libera una herramienta que permite unir las capacidades de su producto Renderman con el poder de modelador Alias|Wavefront[2].

En este año se hacen más intensos los estudios en los modelos de iluminación de objetos mediante la técnica llamada radiosidad. Este modelo se basa en un algoritmo de expansión de calor, por medio del cual se calcula como cada parte de la escena afecta en su iluminación al resto de los objetos. Esta técnica permite crear escenas más reales para ambientes interactivos[FAC97].

Como hemos visto las gráficas por computadora han tenido un gran y rápido desarrollo. Éste se ha debido a que siempre se ha tratado de encontrar la mejor representación posible del ambiente que nos rodea, con el fin de poder comunicar y manipular información de una forma en la que nuestro cerebro la pueda captar y retener más rápido, fácil y por más tiempo.

2.2 Realidad virtual.

2.2.1 Definición e historia.

Aunque existen muchas definiciones acerca de la realidad virtual, todas en esencia coinciden en definirla como:

> *Realidad Virtual.- Es una simulación generada por computadora de un ambiente tridimensional, en la cual uno o varios usuarios pueden ver y manipular los contenidos del ambiente.*

La realidad virtual al igual que muchas otras tecnologías emergió al dominio público después de haber estado por un largo período de tiempo bajo el control de los militares y de unos cuantos centros de investigación académica pertenecientes a algunas universidades estadounidenses [RUD97].

Con su llegada salieron a la luz otras tecnologías como los sistemas de procesamiento gráfico en tiempo real, los dispositivos de alta calidad de despliegue gráfico, la fibra óptica y dispositivos periféricos de control en 3-D. La demanda de los consumidores hizo que esta nueva tecnología poco a poco se fuera perfeccionando y abaratando.

La historia de la realidad virtual comienza en la década de los 1950's cuando un estudiante de cine llamado Morton Heiling escuchó que existía un invento llamado Cinerama, el cual usaba varios proyectores de cine sincronizados de tal forma que los tres eran proyectados hacia un domo, este invento fue creado por Fred Waller durante la Segunda Guerra Mundial y era utilizado como simulador de vuelo por los militares. El cinerama tenía tres pantallas y tres cámaras sincronizada [HAM93].

---

[1] Información recopilada de: http://www.sgi.com, http://www.microsoft.com .

[2] Información recopilada de: http://www.alias.com .

Heilig analizó a detalle los sentidos humanos, llegando a tener un amplio conocimiento de éstos. Ésto lo llevo a construir proyecciones de imágenes tridimensionales sin la utilización de anteojos especiales u otros dispositivos [HAM93].

En 1961 Heiling patenta el Sensorama, que era una máquina en la cual el espectador podía estar inmerso en un paseo en bicicleta. Este aparato proyectaba imágenes 3-D generadas por tres proyectores de cine, sonido estéreo, además de sensaciones olfatorias[1], ésto era realidad virtual sin computadoras. Este aparato se muestra en la figura 2.1 [HAM93].

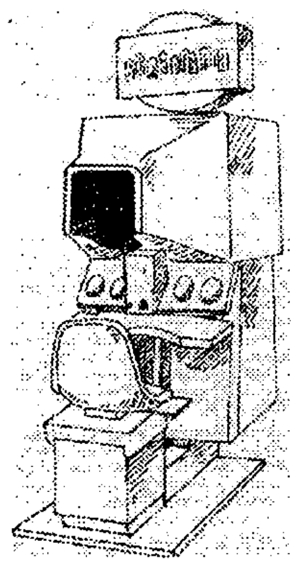


Figura 2.1. Sensorama.

A partir de este momento la realidad virtual empieza a tomar relevancia dentro de los círculos de investigadores en el área de la computación, y comienza a desarrollarse de forma vertiginosa.

1957 Heilig patenta unas gafas (head-monted goggles) las cuales tenían dos unidades de televisión a color.

1960 La compañía Boeing introduce el término de gráficos por computadora.

1962 W.Uttal, de la compañía IBM, patenta un guante para enseñar mecanografía.

1965 Ivan Sutherland publica el libro "The Ultimate Display". El Doctor Frederick P. Brooks comienza a visualizar moléculas de elementos y compuestos químicos, es decir, a crear modelos gráficos en la computadora.

1966 Tom Furnes comienza a trabajar en sistemas de despliegue para pilotos.

[1] Estas sensaciones se producían inyectando varias fragancias en un sistema de aire acondicionado.

1968 Ivan Sutherland publica el libro "A Head-mounted Three Dimensional Display".

Hacia el final de la década de los 1960's, el trabajo de Fred Brooks en la universidad del norte de Carolina exploró el uso de manipuladores mecánicos o control de manipuladores virtuales.

En los 1970's la tecnología raster en los equipos de despliegue gráfico progresó rápidamente gracias a las investigaciones de Gouraud y Phong. Muchos otros contribuyeron a ésto mediante los algoritmos de eliminación de superficies ocultas [VIN95].

1971 Redifon Ltd (Inglaterra) comienza a manufacturar simuladores de vuelo con despliegue de gráficos por computadora. Henri Gouraud presenta su tesis doctoral "Despliegue computarizado de superficies curveadas".

1973 Bui-Tong Phong presenta su tesis doctoral "Iluminación para imágenes generadas por computadora".

1976 P.J. Kilpatrick publica su tesis doctoral "El uso de la kinemática en sistemas gráficos interactivos".

1977 F.H. Raab et al. diseña el sistema de localización Polhemus. Eric Howlett (LEEP systems Inc.) diseña un sistema basado en perspectiva llamado LEEP (Large Expanse Enhanced Perspective).

En la década de los 1980's la realidad virtual fue realmente reconocida como una tecnología viable y con futuro. Las computadoras eran pequeñas, menos caras y rápidas; investigadores en gráficas por computadora estaban produciendo imágenes en grandes cantidades. Sólo eran necesarios algunos años más para que se desarrollaran periféricos tales como guantes y equipos de despliegue de alta calidad [VIN95].

1980 Andy Lippman desarrolla un video disco interactivo que consiste en un paseo por Aspen.

1981 Tom Furness desarrolla el "virtual cokpit", primer mundo virtual interactivo, que consistía en un simulador de vuelo. G.J. Grimes, cede la patente de un guante que permitía la entrada de datos a la computadora a Bell Telephone Laboratories.

1982 Thomas Zimmerman patenta un guante de entrada de datos basado en sensores ópticos, tal que por medio de la refracción interna de éste, se podían correlacionar la flexión y extensión de los dedos.

1983 Mark Callahan construye unos lentes que despliegan imágenes generadas por computadora (Head Mounted Display) para el MIT[1]. Myron Krunger publica el libro "Realidad Artificial".

1984 William Gibson escribe acerca del "cyber espacio" en su libro Neuromancer. Mike McGreevy y Jim Humphries desarrollan el sistema de despliegue de ambientes virtuales VIVED (Virtual Visual Envioroment Display) para entrenamiento de astronautas.

1985 VPL Researchs, Inc. es fundada. Mike McGreevy y Jim Humphries desarrollan un HMD[2] con pequeños dispositivos de despliegue de televisión de LCD (Liquid Crystal Display).

1987 Jonathan Waldern funda industrias W. Tom Zimmerman desarrolla un guante interactivo.

[1] Instituto Tecnológico de Massachusetts.

[2] Head-Mounted Display.

1989 Jaron Lanier inventa el termino "realidad virtual". VLP Research y Autodesk introducen comercialmente los head-mounted displays. Robert Stomne funda el grupo Virtual Reality & Human Factors. Eric Howlett contruye el LEEP video System y HMD. VLP Research,Inc. comienza a vender el Videófono, que utiliza despliegue de LCD y dispositivos ópticos LEEP. Autodesk,Inc. muestra su sistema VR CAD Cyberspace para plataforma PC en SIGGRAPH'89[1]. Robert stone y Jim Hennequin inventan el guante Teletact I. Reflection Technologies produce el HDM Private Eye.
Tomas Furnes funda el HITL (Human Interface Technology Laboratories) en la Universidad de Washington. Estados Unidos.

La década de los 1990's comienza con la fundación de Sense8; Las industria W empiezan a vender sistemas comerciales de realidad virtual para juegos interactivos; Division anuncia la salida al mercado de sistemas de realidad virtual de uso general para el desarrollo y la investigación de la misma [VIN95].

1990 J.R. Hennequin y R. Stone. patentan el guante con dispositivos táctiles. La compañía Sense8 Corporation es fundada por Pat Gelbanb. La compañía Division crea su primer sistema de realidad virtual.

1991 W industries vende su primer sistema de realidad virtual. Richard Holmes patenta un guante con dispositivos táctiles. La compañía Division vende su primer sistema de realidad virtual.

1992 T.G. Zimmerman patenta un guante que utiliza sensores ópticos. Division muestra su sistema multiusuarios de realidad virtual. Nace el World Wide Web.

1993 Silicon Graphics Inc. anuncia la salida al mercado de su sistema Onyx: Reality Engine, que consiste en una máquina diseñada para el trabajo con gráficos por computadora.

1994 InSys y el Manchester Royal Infirmary crean el primer centro de realidad virtual R&D para el diseño de terapias. La sociedad de realidad virtual (Virtual Reality Society) es fundada. Division lanza los Pixel-Planes 5 & 6 desarrollados en la Universidad de Carolina del Norte en Chapel Hill. Un cirujano del hospital Johns Hopkins extirpa un tumor biliar de un paciente, con la ayuda de un HDM. Virtuality anuncia sus sistemas serie 2000. Kaiser Electro-Optics Inc. son contratados por dos años por ARPA, para desarrollar un HDM con un ancho de banda grande. DTI y SERC fundan el proyecto "Virtuosi" para el acceso remoto a lugares virtuales. IBM y Virtuality anuncia el sistema V-SPACE. Division muestra una multiplataforma integrada en sistemas de realidad virtual en el IITSEC de Orlando, Estados Unidos. Nace VRML (Virtual Reality Modeling Language).

1995 Silicon Graphics compra a la compañía de supercomputadoras CRAY-Researchs. Aparece el primer procesador Pentium MMX.

1996 Aparece la versión 2.0 de Cosmo Player, el visualizador de VRML para la plataforma Silicon Graphics.

1997 Tomas Furnes inagura el Laboratorio de Interfaces Inteligentes (HITL[2] - South) en la Facultad de Ingeniería de la Universidad Nacional Autónoma de México (U.N.A.M.).

[1] Convención que se lleva acabo anualmente acerca de las gráficas por computadora.

[2] Human Interface Technology Laboratory.

Cabe mencionar que Tom Furnes dió una conferencia en el auditorio J. Marsal de la división de posgrado de la Facultad de Ingeniería de la U.N.A.M. durante la semana de la realidad virtual en México en Noviembre de 1997. Inaugurando además el Laboratorio de Interfaces Inteligentes (HITL-South), que es una filial del HITL que se encuentra en la Universidad de Washington y que está conformado por dicha universidad y 40 compañías interesadas en el desarrollo y la aplicación de la realidad virtual.

2.2.2 Partes que conforman a los espacios virtuales.

Los sistemas de realidad virtual a grandes rasgos se componen de los siguientes elementos:

1. Sistemas visuales, acústicos y táctiles, por medio de los cuales se busca que el usuario esté más involucrado con el espacio virtual, es decir, que pueda interactuar con él como si estuviera en un ambiente real.

2. Controles de navegación, éstos tienen la finalidad de que el usuario pueda navegar y manipular los objetos del espacio virtual de una forma simple.

3. Procesador central y software. Es necesario elegir las plataformas de hardware y software de acuerdo a la complejidad del espacio virtual que se deseé realizar y de acuerdo a los requerimientos de los usuarios finales, ya que por ejemplo, los intereses de un arquitecto pueden diferir de los intereses de un ingeniero [HAR96].

Es importante mencionar que tanto diseñadores como usuarios de sistemas de realidad virtual, tienen diferentes expectativas de los objetivos que los sistemas deben alcanzar. Para el usuario, un sistema de realidad virtual debe de proporcionarle principalmente 3 tipos de experiencias:

- Manipulación. La posibilidad de alcanzar, tocar y mover objetos en el mundo virtual.

- Navegación. La posibilidad de moverse y explorar el mundo virtual.

- Inmersión. Consiste en aislar al usuario completamente del mundo exterior por medio de dispositivos visuales, táctiles y auditivos [HAR96].

Mientras que para los diseñadores los puntos más importantes de un sistema de realidad virtual son los siguientes:

- Sensación de un ambiente real.- Percepción de la posición de la cabeza, la mano o el cuerpo así como la altura de los usuarios.
- Control del ambiente virtual. Basado en la adquisición de la información sensorial, control de características y comportamiento de los objetos del ambiente.
- Despliegue del ambiente virtual, no sólo el aspecto visual del ambiente, sino también en el aspecto sonoro y táctil del mismo (por mencionar solo unos cuantos de los sentidos humanos) [HAR96].

No solo la ingeniería está participando en el desarrollo de sistemas virtuales, también lo están haciendo ciencias humanísticas como la psicología y la sociología. Para cada una de ellas la realidad virtual tiene otros componentes, por lo que para fines de este trabajo quedan fuera del alcance del mismo.

2.2.3 Estructura de un espacio virtual.

Un espacio virtual se puede definir como:

Espacio virtual.- Es el conjunto de elementos que describen un ambiente de simulación tridimensional, el cual incluye objetos y entidades relacionadas tales como: luces, cámaras, sensores y canales de audio.

Un espacio virtual se compone de:

- Objetos.- Figuras discretas tridimensionales independientes. Pueden ser representadas por elementos visibles, audibles y/o táctiles, lo cual puede crear ideas yuxtapuestas, ésto es, se pueden tocar objetos sin verlos ni oírlos. Pueden estar sujetos a comportamientos predefinidos por el programador o a leyes físicas. Pueden ser estáticos o dinámicos. Objetos estáticos son por ejemplo las paredes de un espacio y dinámicos, por ejemplo un ventilador, ya que independientemente de que interactuemos o no con él, se mueve.

- Fondos.- Es una estructura sobre la cual el ambiente virtual es creado, generalmente son imágenes fijas, aunque en ocasiones pueden tener movimiento, sin embargo éste debe de ser tenue y no muy complejo para no sobrecargar de trabajo al sistema.

- Sensores.- Son elementos que pueden formar parte de un espacio virtual, éstos generan posición, orientación y otros datos, leyendo las entradas que se originan en el mundo real proporcionadas por la interacción con el usuario, permitiéndole a éste la manipulación de objetos dentro del espacio virtual.

- Cámaras.- Permiten al observador tener una perspectiva del espacio virtual, generalmente nos proporcionan una salida visual a diferentes dispositivos (monitor, casco, lentes, etc.).

- Luces.- Se utilizan para aumentar la intensidad de iluminación en ciertas partes del ambiente. Existen diferentes tipos de luz : direccionales, de área, puntuales, ambientales , etc. . Las intensidades de éstas así como los colores pueden cambiar.

Los objetos que no se encuentran dentro de la vista de la cámara no existen para la aplicación, hasta el momento en que ésta los toma, lográndose con lo anterior la construcción de espacios virtuales muy complejos [DEZ97].

La calidad de los espacios virtuales puede variar de acuerdo a las plataformas de hardware y software que se estén empleando. Las principales limitaciones técnicas para tener espacios virtuales de gran calidad radican en que el hardware no es lo suficientemente rápido para estar realizando todas las operaciones matemáticas que requeriría el tener una escena de alta calidad. Por el lado del software lo que nos limita es la representación de los objetos y los algoritmos que se tienen para el mapeado de texturas así como para el "render". El render es un algoritmo que permite mapear colores, brillos y texturas a los objetos de una escena. Mientras más calidad de render se deseé mayor será el número de cálculos que se deberán de llevar a cabo. Por otra parte los objetos son modelados mediante polígonos, lo cual implica que para crear superficies o líneas curvas se necesita una gran cantidad de ellos, ésto se traduce en una cantidad muy grande de datos que hacen que la geometría se muy "pesada" para el hardware, perdiendo con ésto velocidad. Debemos de recordar que un sistema de realidad virtual debe de ser procesado en tiempo real, ya que de otra forma el usuario no tendrá la sensación de inmersión, por lo que perderá el interés en la aplicación.

Debemos recordar que dentro de la realidad virtual influyen factores psicológicos, que muchas veces hacen que el usuario que se encuentra interactuando con un espacio virtual no necesite de gráficos de alta calidad para sentirse inmerso en el mundo, sino que sólo le baste con ver objetos con textura plana de no mucho detalle, sin embargo, puede ocurrir también todo lo contrario, por lo que la calidad del ambiente virtual siempre estará a criterio del diseñador, quien a fin de cuentas es quien sabe mejor cuáles son las necesidades de su usuario.

2.2.4 Dispositivos periféricos utilizados.

Dentro de los sistemas de realidad virtual existe una gran variedad de dispositivos, cuyo fin es proporcionar al usuario un estado de inmersión completo, ésto es, que sienta que está en un ambiente lo más real posible. Podemos clasificar a estos dispositivos en dos grandes grupos [DES97] :

1. Dispositivos de entrada.

2. Dispositivos de salida.

Los dispositivos de entrada a su vez se clasifican en dos tipos:

- Dispositivos de localización.

- Dispositivos de control.

Los dispositivos de localización tienen como objetivo detectar la posición y la orientación del usuario, así como permitir la navegación del mismo dentro del espacio virtual. Estos dispositivos tienen que implementarse de tal forma que se adapten al ser humano y no el ser humano se tenga que adaptar a ellos. Ya que mientras más sencillos sean de manipular, el usuario navegará de una forma más natural dentro del espacio virtual.

Los dispositivos de localización se dividen, a grandes rasgos en:

- Electromagnéticos.- Utilizan un transmisor de señales radioeléctricas que emite a intervalos regulares tres señales electromagnéticas cuyos campos son ortogonales entre si. Dichas señales se reciben en un receptor formado por tres bobinas perpendiculares y se comparan para deducir la posición y orientación que los receptores tienen con respecto al emisor. El emisor y el receptor se conectan a un equipo electrónico que realiza los cálculos de la posición y orientación. Su alcance es limitado y el error se incrementa al aumentar la distancia entre el emisor y el receptor, lo mismo sucede si existe algún objeto metálico cerca del equipo o si éste se encuentra cerca de algún equipo electrónico (como monitores) o de equipo eléctrico como las balastras que se utilizan en las lámparas de neón. Su tiempo de respuesta en ocasiones suele ser alto. Sin embargo permite libertad de movimiento dentro de un rango aceptable y su costo es bajo. Existen varios fabricantes de este tipo de dispositivos, los más reconocidos son Polhemus y Ascensión Technologies. En la figura 2.2 podemos ver uno de los dispositivos electromagnéticos de localización fabricado por Polhemus.

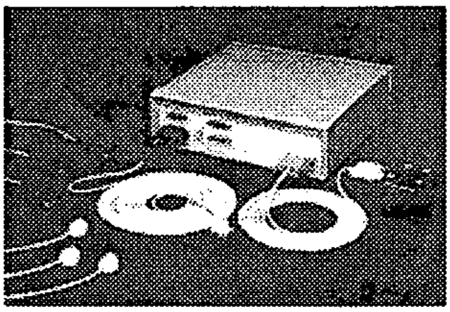


Figura 2.2. Dispositivo Electromagnético de la compañía Polhemus modelo Fast Trak .

- Mecánicos.- Utilizan un brazo con múltiples articulaciones para determinar la orientación y posición del usuario. Uno de los extremos del brazo se encuentra fijo a un cierto punto, mientras que otro está adherido al dispositivo de visualización o a la cabeza del usuario. Como la longitud de los elementos del brazo articulado es fija, sólo se tiene que medir los ángulos de cada una de las articulaciones (con sensores ópticos o de resistencia variables) para conocer con exactitud la posición y orientación del extremo. Este sistema es muy preciso y rápido, sin embargo el movimiento del usuario está limitado al alcance y flexibilidad de las articulaciones del sistema. Existen varias compañías como Fake Space

Labs que fabrican este tipo de dispositivos. En la figura 2.3 tenemos un ejemplo, generalmente es llamado "boom".

Figura 2.3. Dispositivo Mecánico de localización (*boom*).

- Ultrasónicos.- Para la localización en este tipo de sistemas, se emplea un procedimiento de triangulación. Un circuito emisor produce una onda ultrasónica que es recogida por tres receptores dispuestos triangularmente. Cuando el emisor produce una señal, se dispara un contador en los tres receptores, se mide el tiempo que la señal tarda en llegar hasta cada uno de ellos, obteniendo así la posición absoluta del emisor. Sin embargo, necesita una línea directa de visión entre emisor y receptor; es un sistema bidireccional (el emisor debe estar dirigido hacia los receptores). Este sistema se utiliza en ratones 3D.

Aparte de estos dispositivos existen otros que aún se encuentran en etapa de investigación [DEZ97].

Por otra parte los dispositivos de control permiten al usuario tener un dominio explícito de las aplicaciones e interactuar con el espacio virtual a través de órdenes, las cuales pueden ser:

- Comandos de navegación .- Éstos son ordenes, mediante las cuales el usuario indica al sistema que desea desplazarse dentro del espacio virtual.

- Comandos de interacción.- Son las ordenes que permiten mover, usar o manipular elementos que conforman el espacio virtual.

- Comandos de manipulación.- Permiten la modificación de las propiedades y del comportamiento de los objetos dentro del espacio virtual.

Dentro de estos dispositivos encontramos a los guantes de datos, que se encargan de sensar la posición de la mano del usuario en los espacios virtuales, pudiendo detectar hasta el movimiento de los dedos [DEZ97]. En la figura 2.4 podemos ver uno de estos dispositivos.

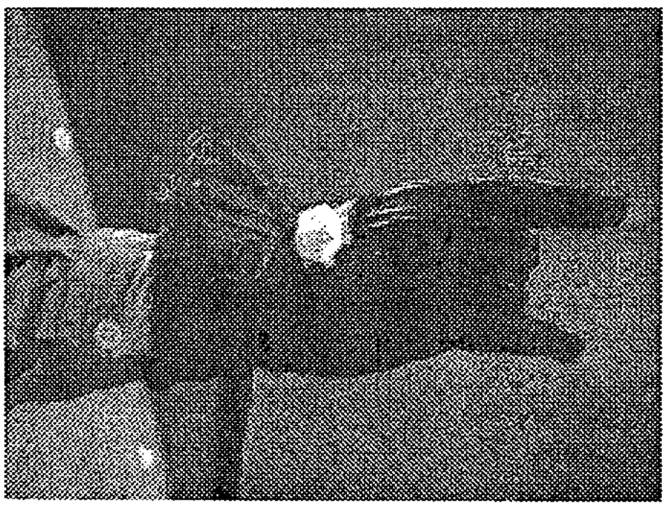


Figura 2.4. Guante de datos.

Los trajes funcionan prácticamente igual que el guante de datos, sólo que cubre todo el cuerpo. Se elaboran con fibra óptica, ésto con el fin de poder monitorear partes del cuerpo por medio de las interrupciones de la luz en la fibra óptica. La empresa Virtual Technologies fabrica el CyberSuit, un traje completo para detectar la postura en la que el usuario se encuentra. En la figura 2.5 se muestra este dispositivo.

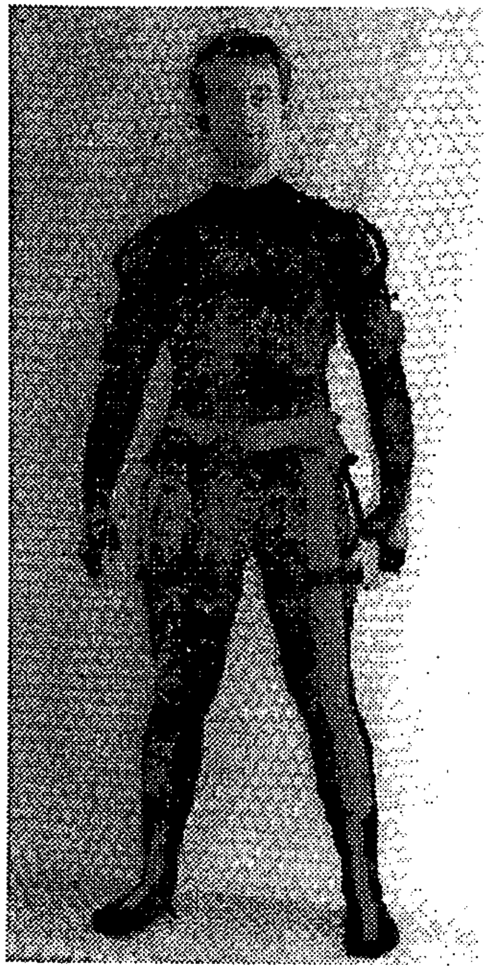


Figura 2.5. Traje (body).

Los ratones 3D son una extensión natural de los ratones convencionales, ya que permiten controlar el movimiento de cualquier objeto en un espacio tridimensional. Existen diferentes tipos de ratones 3D, cada uno de los cuales utiliza un método de localización que puede ser cualquiera de los anteriormente mencionado [DEZ97]. En la figura 2.6 podemos ver el mouse 3-D de DIVISION.

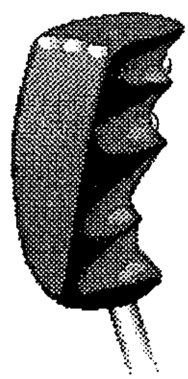


Figura 2.6. Mouse tridimensional (o joistick) de DIVISION.

Existen además sistemas bioeléctricos que miden señales eléctricas generadas por el cuerpo humano y las transforma en señales de control analógicas y digitales que pueden emplearse para gobernar cualquier tipo de dispositivo. Por ejemplo, la empresa Biocontrol Systems ha desarrollado un control ocular llamado Eyecon que detecta y procesa las señales generadas por los movimientos del ojo (electro-oculograma) y las transforma en señales de control que pueden utilizarse en cualquier aplicación.

Otro dispositivo llamado Brainman fabricado por Biocontrol System, permite detectar y transformar en señales de control las ondas cerebrales. El problema con este dispositivo es que las ondas cerebrales no pueden controlarse de forma consciente, lo cual hace que este tipo de dispositivos sean poco prácticos además de costosos. Por otra parte existen dispositivos de reconocimiento de voz e imágenes que permiten ejecutar comandos vocales y comandos activados por gestos [DEZ97].

Los dispositivos de salida por su parte, permiten que el usuario tenga una visión[1] adecuada del espacio virtual. Se pueden clasificar de forma sencilla de acuerdo al sentido del usuario que se pretenda estimular, bajo este criterio los podemos agrupar en:

- Dispositivos de visualización o presentación. Tienen por objetivo mostrar al usuario la información de carácter gráfico. Éstos constan de unos lentes con dos dispositivos de despliegue, que generalmente son pantallas LCD, como el de la figura 2.7.

Figura 2.7. Head-mounted display de la compañía Visual Research.

- Dispositivos de audio. Están encargados de generar la información sonora acerca del espacio virtual. Generalmente consisten de dos audífonos que van incluidos en los cascos de realidad virtual. Figura 2.7.

[1] Entendiendo por visión a todos los sentidos, no únicamente al sentido de la vista.

- Dispositivos de realimentación táctil. Simulan las propiedades de los materiales que el ser humano percibe a través del sentido del tacto, por ejemplo, la rugosidad de un objeto.

- Dispositivos de realimentación cinestésica. Utilizados para simular propiedades mecánicas de los objetos, tales como la resistencia o la inercia.

- Dispositivos móviles. Permiten simular movimientos a través del espacio virtual.

2.3 Campos de aplicación.

En la actualidad las aplicaciones de la realidad virtual son muy variadas, incluyen comercio, entretenimiento, diseño de productos y medicina, por mencionar algunos [VIN95].

En la ingeniería ayuda a la creación de modelos de piezas y/o mecanismos cuyo costo de construcción sería muy elevado. Además de permitir la manipulación de equipo costoso que puede ser simulado virtualmente, con lo que se evita el deterioro del equipo real, por ejemplo en el caso de los robots.

En la industria aeronáutica se emplea para modelar turbinas, controles y para el diseño de los interiores de los aviones. En el caso de las turbinas, la realidad virtual permite hacer modificaciones a las piezas, ésto con el fin de perfeccionar su funcionamiento. Una vez terminado el diseño, se manda construir el avión, sin necesidad de haber invertido gran cantidad de recursos en el desarrollo del mismo.

En la arquitectura, facilita el diseño de las construcciones y elimina la costosa tarea de hacer maquetas prototipo del proyecto. Ésto permite que los arquitectos se centren en la idea de la creación de la obra y no en como y con que materiales hacer la maqueta. Por otra parte, una vez terminado el modelo se puede recorrer como si el proyecto ya estuviese construido.

La realidad virtual también es empleada para modelar objetos, tales como automóviles y/o máquinas caseras, con el fin de hacerlas más ergonómicas.

En la industria automotriz, se pueden construir los modelos de los automóviles y ponerlos a consideración del consumidor o del empresario, de tal forma que ellos puedan obtener un diseño que les agrade.

Otros nuevos usuarios han sido la compañías de telecomunicaciones, las cuales emplean la realidad virtual para el diseño de la instalación de redes tanto telefónicas, como de computadoras o de cualquier otro tipo de servicio que ofrezcan.

Sin duda el uso que más ha impulsado la comercialización de esta tecnología ha sido el entretenimiento, ésto debido a que es la actividad en la que la gente siempre está dispuesta a invertir. En la actualidad podemos apreciar como cada vez son más los anuncios comerciales que incluyen gráficas tridimensionales. La meta es llegar a tener anuncios interactivos, en donde el consumidor pueda apreciar y manipular el producto. Un claro ejemplo de ésto lo tenemos en las compañías que utilizan Internet para hacer publicidad; por ejemplo Barcadi Inc. tiene en su pagina de Web[1] un bar virtual en el cual uno puede ver publicidad de Bacardi así como interactuar con otros "cyberasistentes" que se encuentran en ese sitio. Este bar fue construido con VRML. También existen juegos que emplean la realidad virtual para hacer sentir al usuario que él está protagonizando la historia.

En la televisión mexicana durante las elecciones del jefe de gobierno del Distrito Federal pudimos apreciar como las compañías televisoras (Televisa y Televisión Azteca) utilizaban "sets virtuales" para mostrar los resultados parciales de las elecciones. Estas compañías utilizan la realidad virtual para baja costos en la escenografía de los programas de televisión, ya que aunque la calidad de los escenarios no es muy alta da una buena aproximación de un escenario real.

En la ciencia, la realidad virtual auxilia a la medicina, la astronomía y la química creando modelos de los objetos de interés para cada una de ellas.

El entrenamiento de personal es quizá la actividad en la cual la realidad virtual es más empleada. La NASA[2] ha utilizado espacios virtuales para dar entrenamiento a astronautas en reparaciones de equipo en el espacio. Los bomberos comienzan a utilizar ambientes virtuales en donde entrenan como apagar incendios. El ejemplo clásico de el entrenamiento mediante realidad virtual son los simuladores de vuelo, en donde los pilotos aprenden como volar un avión, ahorrándose con ésto grandes cantidades de dinero. Los militares también utilizan espacios virtuales para su entrenamiento tanto en infantería como en el diseño de estrategias de combate [VIN95].

La plantas nucleoeléctricas utilizan consolas de control de reactores virtuales para dar capacitación a los técnicos que tendrán la responsabilidad de operarlos. Finalmente podemos decir que la realidad virtual es empleada para la prevención de accidentes y para la creación de planes de contingencia durante catástrofes naturales.

2.4 Estandarización.

Como hemos visto, la realidad virtual es una tecnología que debido a sus características proporciona muchos beneficios a toda la humanidad. Sin embargo en nuestros días es muy difícil llevarla al alcance de todas las personas, ya que el costo del hardware y del software de realidad virtual sigue siendo muy alto.

Ésta fue una de las motivaciones que llevaron a muchos investigadores a reunir esfuerzos par poder crear soluciones baratas y accesibles a todo público. Estos esfuerzos culminaron en la creación del lenguaje de realidad virtual VRML (Virtual Reality Modeling Language).

VRML fue concebido como un estándar, por lo que podría correr en cualquier tipo de plataforma. Por otro lado como se pretendía que cualquier persona tuviera acceso a esta aplicación, se pensó incluir visores de VRML a los browsers ó navegadores de Internet, con lo que estaría al alcance de cualquier persona.

[1] La dirección es: *http://www.lunatic.de/en/wow/bacardi/bar1.htm* .

[2] National Aeronautical and Space Administration.

La programación en VRML se lleva acabo mediante un lenguaje compuesto de primitivas y directivas, es decir, es interpretado y no compilado. Para poder dar un poco más de versatilidad a VRML se implementó la comunicación con el lenguaje Java.

Varias compañías apoyaron y siguen apoyando a este lenguaje, entre ellas encontramos a Microsoft y a Silicon Graphics.

Aunque VRML puede en teoría correr en cualquier plataforma, existen todavía algunos problemas cuando se trata de correr un mundo virtual en una PC o en algunas estaciones de trabajo, por ejemplo en las estaciones Sparc de SUN.

En el caso de las estaciones de trabajo Silicon Graphics, VRML está basado en las bibliotecas gráficas Open Inventor, las cuales permiten un despliegue y una manipulación óptima de objetos tridimensionales. Sin embargo Open Inventor sólo corre bajo plataforma Silicon Graphics. Por lo que viendo las ventajas que ofrecen estas bibliotecas se han comenzado a estandarizar también.

Finalmente cabe mencionar que la estandarización no es una solución fácil, sin embargo permite que tecnologías como la realidad virtual estén al alcance de todas las personas.

Capítulo 3.

Diseño de espacios virtuales.

El diseño de ambiente virtuales consiste de tres etapas:

- Preparación
- Solución
- Salida

Ésto lo podemos ver de manera más detallada en la figura 3.1. [DEZ97]:

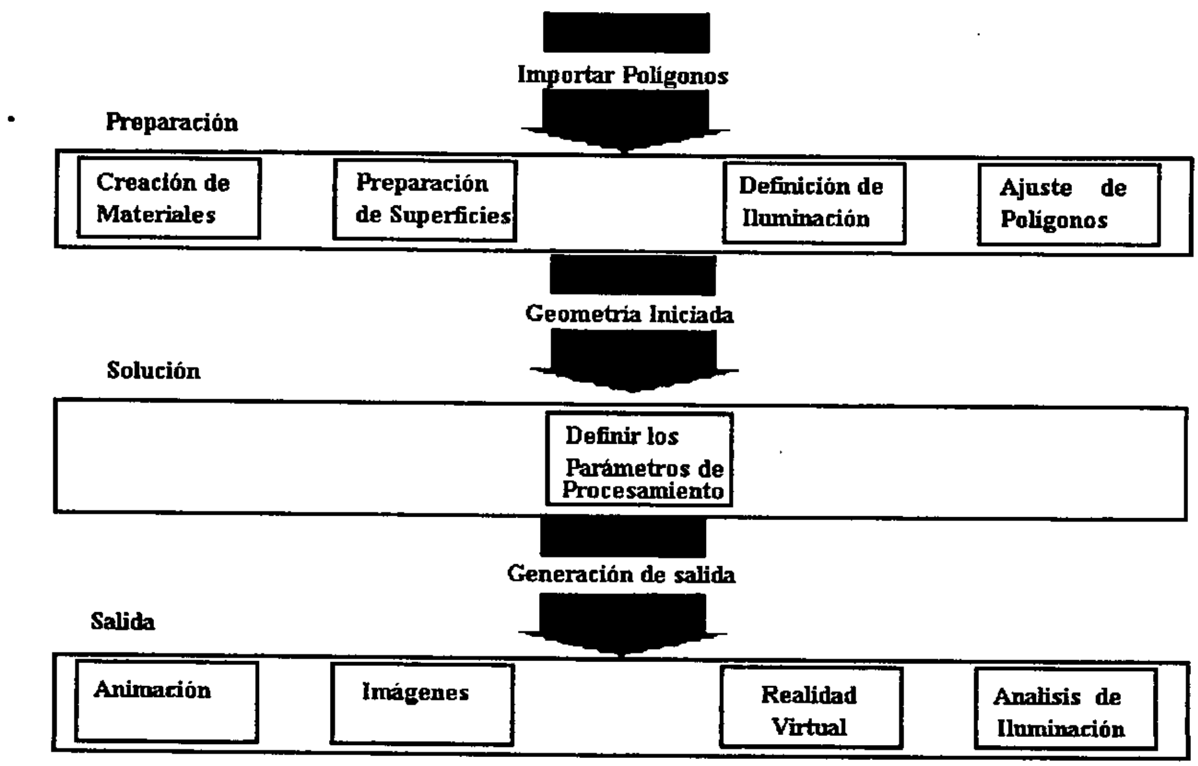


Figura 3.1. Pasos para la creación de un espacio virtual.

Para cada una estas etapas debe tenerse presente:

- El punto de vista del usuario en primera persona, ésto es, se le debe permitir realizar movimientos completos a voluntad en tiempo real.

- La capacidad de visualizar y modificar el entorno virtual en tiempo real [GRA97].

De acuerdo con el Dr. David Seltzer del MIT, se considera que todo espacio virtual debe incluir despliegue visual, audio tridimensional, tacto, percepción de la posición y de la actividad motora, es decir, un buen marco de referencia, el cual se compone de:

- Modelos matemáticos.

- Paradigmas de interacción.

- Interfaces lógicas con el modelo computarizado

- Interfaces fisicas [DEZ97].

Antes de comenzar a modelar el espacio virtual con un programa de CAD[1], es necesario tener una idea clara de lo que se desea representar. Para ésto es recomendable hacer dibujos en papel, tratando de describir de una manera general cuáles van a ser los componentes el espacio virtual [DEZ97].

Un aspecto importante a considerar para la elección del modelador, es el tipo de formato en el que nos va a guardar el modelo, es decir, el formato del archivo de salida. Se recomienda que la información siempre se almacene en un formato que sea lo más estándar posible[2].

Una vez terminado el modelado, se procede a la elección de los materiales del espacio virtual. Los materiales son las características visuales que deseamos que nuestros objetos tengan, es decir, el color y la textura. Se recomienda elegirlos de acuerdo a las características de los objetos reales, ésto ayudará a dar mayor realismo a nuestro espacio virtual. Las texturas son imágenes de la superficie de los objetos, por ejemplo pisos o tabiques, los cuales se mapean en la geometría, proporcionándole una mayor detalle a los objetos. En esta etapa se recomienda optimizar la geometría del modelo. Esta labor consiste en reducir la cantidad de polígonos del modelo tanto como sea posible o en su defecto ordenar la información de tal forma que sea menos pesada para el hardware.

Debemos de asegurarnos que las caras de los objetos estén orientados en la dirección correcta, debido a que los algoritmos de iluminación se basan en las normales de las superficies que forman a los objetos. Si alguna de las caras de un objeto está mal orientada aparecerá obscura y opaca en el espacio virtual o incluso puede no aparecer. También se deben definir las propiedades de las luces que van a ser utilizadas en el espacio virtual, ésto consiste en ajustar el color e intensidad de cada una de ellas.

---

[1] Computer Aided Desing (CAD).

[2] En la mayoría de las aplicaciones de CAD el formato DXF es reconocido, por lo que se puede considerar como un buen formato de almacenamiento.

Antes de comenzar con la etapa de solución se recomienda hacer todos los cambios al modelo, ésto nos ahorrará retrasos y modificaciones tediosas.

Una vez ya preparada la geometría, entra a la etapa de Solución. En esta etapa se le agregan características y comportamientos al modelo. Ésto consiste en el refinamiento de colores y texturas. En esta parte del proceso se emplean paquetes comerciales que permiten visualizar imágenes preliminares del modelo, generalmente son imágenes estáticas.

Con lo anterior lo que se logra es pulir el mapeado de texturas y la iluminación. En esta etapa es muy común cargar demasiado el modelo, es decir, se le ponen texturas que son muy grandes y por consiguiente pesadas de manejar para la computadora. Por otra parte si tenemos pocos polígonos, las texturas se pueden ver muy "pixeleadas"[1] y tendremos imágenes de poca calidad. Es entonces cuando se tiene que hacer un análisis de que es lo más conveniente, es decir, si queremos mucha calidad de imagen pero poca velocidad o viceversa. Lo óptimo es encontrar un punto medio, en el cual la geometría no sea muy pesada, pero tampoco sea muy pobre, para que la calidad de las imágenes sea aceptable.

En el caso de la iluminación se recomienda no poner demasiadas luces y no asumir que éstas se comportan de forma similar a las fuentes de luz natural. El poner muchas luces hace que la imagen de salida además de ser pesada para la computadora, se vea demasiado blanca y opaca, perdiendo detalle. Debemos de recordar que los algoritmos de render se basan en el cálculo de la iluminación de cada punto de la imagen, por lo que a cada fuente de luz se le aplica este algoritmo.

La iluminación en un espacio virtual no es calculada tomando en cuenta todas las propiedades físicas de la luz, sólo se considera el choque con los objetos que tiene enfrente la fuente de luz (algoritmo de ray tracing). Por otra parte se recomienda no utilizar colores puros[2] ya que éstos producen un efecto llamado "sangrado" en los dispositivos de despliegue. Este efecto consiste en que el color es esparcido por áreas más extensas de las delimitadas por el objeto, es entonces cuando se ven fantasmas en las imágenes o límites de objetos muy poco definidos.

Existen muchos paquetes comerciales que nos permiten agregar texturas y modificar atributos de color e iluminación en un modelo alambrado.

Finalmente en la etapa de salida es en donde se definen las posiciones de las cámaras, así como el tipo de dispositivos de despliegue que se van a utilizar. Se conjuntan tanto imágenes como sonido. Aunque no se ha hablado de sonido, éste tiene un procesamiento un poco más sencillo que el de las imágenes. El sonido se captura mediante herramientas de software y por medio de tarjetas especiales. Una vez almacenado se puede editar su duración, su volumen y su timbre (graves o agudo), además de tener la posibilidad de agregarle efectos como ecos y deformaciones. Posteriormente se introduce el sonido al espacio y se decide mediante que acción se va a activar, por ejemplo, cuando se deja caer un objeto en el piso.

Por otra parte puede ser que el modelo alambrado del espacio virtual se quiera mostrar por medio de una animación, imágenes estáticas o introduciéndolo a un sistema de realidad virtual. En el caso de la animación es necesario predefinir una trayectoria de la cámara para simular una navegación dentro del espacio virtual. En el caso de imágenes estáticas, sólo basta con elegir una vista del modelo y tomarle una "fotografía", es decir, "renderear" esa vista y darle algún formato de imagen (jpeg, gif, tiff, rgb, etc.). Cuando se introduce a un sistema de realidad virtual, basta con aplicar los convertidores adecuados a la geometría[3] para que

[1] El termino pixeleado se refiere a que en una imagen el tamaño del pixel es muy grande y por lo tanto la calidad visual de la misma es muy pobre.

[2] Un color puro, es un color que no tiene ninguna otra componente de otro color primario, por ejemplo un rojo que no tiene ninguna componente de verde y azul.

[3] Geometria es el nombre con el que se hace rerefencia al modelo alambrado del espacio virtual.

pueda ser reconocida por el paquete. En el caso del análisis de iluminación, lo que se busca es encontrar los parámetros y los algoritmos adecuados para producir escenas lo más real posible. Existen algunos algoritmos como el de radiosidad que dan una muy buena calidad de iluminación. Sin embargo tienen la desventaja de no poder producir brillos metálicos y reflexión de la luz en superficies, además de no permitir la interacción con los objetos del mundo virtual. Actualmente se está trabajando en encontrar algoritmos que brinden las ventajas del método de radiosidad y del método de ray tracing, con lo que se obtendrían imágenes muy reales que pudieran ser producidas en tiempo real. Por último cabe mencionar que el método de creación de espacios virtuales visto en este capítulo puede tener otras variantes, ésto dependerá de las necesidades de cada usuario y de los recursos tanto económicos como de cómputo de los que se disponga .

Capítulo 4.

Selección de herramientas para la construcción de espacios virtuales.

Como ya vimos en el capítulo anterior, existen diferentes etapas en el desarrollo de un espacio virtual. Por lo que en cada una de ellas las necesidades de software y hardware son diferentes.

Para la creación del modelo alambrado del espacio virtual de nuestro caso de estudio, se decidió emplear el modelador Alias|Wavefront, debido a que proporciona grandes facilidades en el modelado de cualquier tipo de objeto, permitiendo guardar la información en una amplia variedad de formatos, los cuales son compatibles con paquetes para PC y para UNIX. Alias|Wavefront corre bajo UNIX en máquinas Silicon Graphics. Este paquete se encuentra en el Laboratorio de Visualización del departamento de supercómputo de la Dirección General de Servicios de Cómputo Académico (D.G.S.C.A.) de la U.N.A.M.

Para la etapa de solución[1] se decidió emplear el paquete para PC "3D-Studio". Ésto debido a que proporciona muchas facilidades para el mapeo de texturas y para la creación de materiales. Por otra parte reconoce el formato DXF[2], con lo que podemos manipular modelos creados con Alias|Wavefront. La razón por la que no se usa 3D-Studio para modelar objetos es debido a que es un paquete para edición de imágenes y animaciones, por lo que las herramientas que proporciona para el modelado son muy sencillas y poco potentes, por lo cual no es adecuado para crear modelos alambrados. Por otra parte el formato de archivo de 3D-Studio es muy compacto, por lo que genera archivos pequeños, además existen muchos convertidores de formato para este paquete, haciéndolo muy versátil.

Para la etapa de salida se eligieron tres herramientas de realidad virtual:

- El paquete "dVISE" de la compañía DIVISION (dvreality versión 4.0.1).

- El visualizador de VRML "Cosmo Player" de Silicon Graphics (versión 1.0.1).

- Las bibliotecas gráficas Open Inventor de Silicon Graphics (Programación en ASCII versión 2.1.3).

A continuación se describe cada una de ellas:

DVISE.

DVISE es una herramienta de simulación y construcción de espacios virtuales. Está basada en un modelo simple en el cual un mundo virtual está compuesto de un cierto número de objetos, cada uno con sus propios atributos tales como color, movimiento y características de audio que son predefinidas por el usuario. Los objetos pueden ser modelados con una amplia variedad de paquetes comerciales de CAD y después importados a dVISE [VIN95].

---

[1] Ver figura 3.1.

[2] Data eXchange inFormation.

DVISE permite modificar las propiedades del espacio virtual por medio de una interfaz de ventanas, con las que el usuario modifica su modelo y puede apreciar inmediatamente los cambios hechos. Otra forma de hacer ésto es mediante una herramienta llamada "tool box", que consiste en una serie de menús tridimensionales con los que también se pueden modificar las propiedades del espacio virtual, estando inmerso en él, es decir, desde dentro.

Los archivos de dVISE tienen un formato de salida llamado MAZ y un formato ASCII, el cual puede ser modificado utilizando cualquier editor de texto. Además cuenta con convertidores de formato, éstos transforman archivos en formato DXF o 3DS[1] (entre otros) a formato MAZ, proporcionándole a dVISE una gran versatilidad.

Las instrucciones dentro de los archivos MAZ son utilizadas para el control de detección de colisiones, iluminación, acústica, actualización de la imagen, tamaño, posición y orientación de los objetos[2].

DVISE está escrito en lenguaje C, por lo que los usuarios pueden adecuar aún más el paquete a sus necesidades, por medio de la creación de rutinas que le den algún comportamiento especial a los objetos [DIV97].

El objetivo principal en el diseño de dVISE fue el crear un modelo distribuido para la ejecución de realidad virtual. Este esquema simplifica el proceso de definir complejos espacios virtuales, además de mejorar el desempeño de las aplicaciones. Un modelo paralelo de interfaz de usuario 3D es muy conveniente para ayudar a simplificar el control de todos los elementos que componen un sistema de este tipo, ésto produjo un poderoso paradigma de desarrollo [DIV97].

En este modelo distribuido, existen varios servidores o "actores" que se encargan de realizar diferentes tareas o funciones. Un actor se define como un programa que está encargado de una sóla función específica en el sistema, y de forma conjunta con otros actores, hace posible la ejecución del espacio virtual. Las funciones de un actor se pueden dividir en tres grupos :

1. Obtención de datos del mundo real, es decir, se encargan de obtener información de los diferentes dispositivos periféricos utilizados en el sistema.

2. Control del espacio virtual, se encargan de mover los objetos así como de activar el sonido, de acuerdo a la información proporcionada por los dispositivos periféricos del sistema.

3. Despliegue del espacio virtual, se encargan tanto del despliegue visual como acústico y táctil del espacio virtual.

La figura 4.1 muestra las relaciones entre los diferentes actores. El actor de despliegue (Visual) muestra los objetos en pantalla, el actor de audio(Audio) se encarga del sonido, el actor de colisiones(Collide) detecta cuando dos objetos están en contacto y no permite que los atravesemos. El Body Actor nos proporciona una representación del usuario en el espacio virtual. Finalmente el actor de rastreo (Input) nos da la posición 3D que tiene el usuario en un momento determinado [DIV97].

[1] Formato de 3D-Studio.

[2] Para mayor información de este formato, consultar los manuales que proporciona el fabricante.

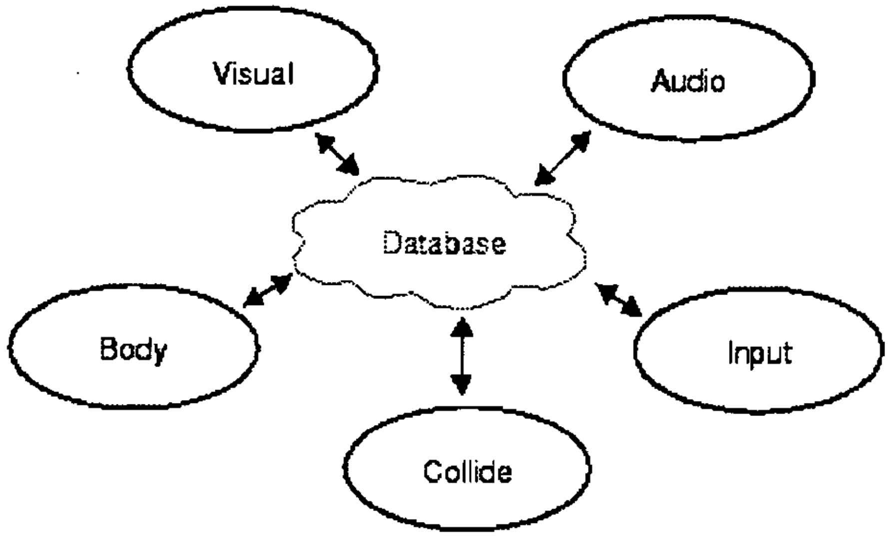


Figura 4.1. Esquema de comportamiento de los actores.

El modelo anteriormente descrito permite tener una interfaz eficiente. En el esquema tradicional de las arquitecturas cliente servidor, un conjunto de mensajes son definidos para encapsular lo más posible las funciones de un servidor, entonces el cliente tiene que enviar las peticiones al servidor mediante llamadas locales o remotas. Sin embargo ésto no es útil en sistemas multiusuarios, en los cuales diferentes partes de una interfaz son capturadas por diferentes servidores. Lo cual se complica aun más si estos servidores necesitan intercambiar información entre ellos.

Para solucionar este problema dVISE adoptó un modelo más abstracto de objeto. Éste consiste en que cada actor establece una conexión directa con los otros actores mediante un área compartida de datos en la cual cada actor coloca objetos llamados "elementos". Los actores monitorean estos "elementos" y responden de forma global a los cambios hechos en el espacio virtual mediante este mecanismo. Los actores se comunican a través de estos objetos compartidos, ésto se traduce en una alta funcionalidad del sistema. El acceso a esta área compartida es provista por las rutinas de la biblioteca llamada VL, que a su vez utiliza los semáforos proporcionados por el sistema operativo UNIX para el control de acceso a regiones críticas. Estas funciones permiten crear, borra, extraer y modificar elementos e instancias en el sistema. El ambiente es, en efecto, la base de datos de VL [VIN95].

Para terminar con dVISE, cabe mencionar que éste poseé otra herramienta llamada VCToolkit, que es una colección de funciones de alto nivel en lenguaje C (a diferencia de las funciones de VL que son de bajo nivel), las cuales permiten manipular objetos virtuales. Ésto proporciona al usuario la facilidad de poder programar y crear objetos virtuales en la forma que él decida pudiendo ser jerárquica. Estas funciones pueden manipular también las propiedades visuales, acústicas y físicas de los objetos. También existen funciones que permiten crear, modificar y borrar elementos estándar del sistema, además de agregarles funciones de comportamiento llamadas callbakcs, las cuales generalmente se activan cuando ocurre un evento en el programa, por ejemplo cuando se presiona un botón o cuando se da click en una ventana. Los eventos a los que reaccionan las funciones del Toolkit son capturados por el modulo Vsrun-time [VIN95].

Cabe mencionar que además del software, los productos de DIVISION incluyen algunos dispositivos que sirven para capturar señales emitidas por los periféricos empleados en el espacio virtual. Esta unidad es conocida como IPU (Input Procees Unit).

La descripción de los actores, así como su configuración se cubren ampliamente en los manuales que proporciona DIVISION con su software.

VRML.

VRML es un lenguaje de programación para realidad virtual, el cual puede ser desplegado por varios visores. Un visor es un programa que permite navegar a través de los espacios virtuales construidos con VRML. El visor de nuestro interés es Cosmo Player. Este visor ofrece todas las ventajas del hardware de Silicon Graphics, debido a que está programado de tal forma que aprovecha todos los recursos de esta plataforma. Se puede obtener de forma gratuita en Internet[1]. Se instala como un aditamento(plug-in) para el browser (navegador de Internet), lo cual permite que al visitar paginas que contengan espacios virtuales, éstos se carguen de forma automática [SIL397].

Una característica de VRML es que es un lenguaje interpretado, por lo cual es portable a cualquier plataforma, conviertiendose así en un estándar. VRML puede conectarse al lenguaje de programación Java, con lo que pueden hacerse espacios virtuales con comportamientos complicados.

VRML puede tener conexión con otros programas mediante el uso de sockets[2] programados en Java, con lo que su potencial aumenta. En VRML los espacios virtuales se construyen mediante primitivas y directivas que se guardan en archivos con extensión ".wrl" [RUD97].

Cabe señalar que existen una gran variedad de convertidores de formato para VRML, éstos también se pueden obtener de forma gratuita en Internet. 3D-Studio puede salvar modelos en el formato de VRML.

Open Inventor.

Open Inventor es un conjunto de bibliotecas gráficas mediante las cuales se pueden construir objetos tridimensionales y por consiguiente espacios virtuales. Ofrece una serie de funciones para los lenguajes C y C++, además de tener una programación ASCII, es decir, se pueden programar objetos mediante directivas y primitivas. Éstas se almacenan en un archivo con extensión ".iv". Estos archivos pueden ser vistos mediante el uso del visor *ivview(1)*, el cual también permite la navegación en dichos espacios virtuales. Al igual que para VRML, existen muchos convertidores (gratuitos) de otros formatos a Open Inventor en formato ASCII. Cabe mencionar que VRML en su versión 1.0 es compatible con Open Inventor, es decir, no se necesita de ningún convertidor para ver mundos de VRML 1.0 en *ivview* o para ver un espacio virtual hecho en Open Inventor con Cosmo Player [WER94].

Open Inventor combina el uso de los recursos de cómputo gráfico que ofrece Silicon Graphics con el estándar de ventanas X11, lo cual permite la creación de ambientes virtuales muy complejos [WER94]. Tanto dVISE como Cosmo Player utilizan las funciones de Open Inventor para el levantamiento de los espacios virtuales.

Por el lado del hardware se eligió la plataforma Silicon Graphics, debido a que este tipo de arquitectura proporciona tarjetas con microprocesadores dedicados exclusivamente al procesamiento de gráficas y sonido, todo ésto bajo el sistema operativo UNIX.

De las tres herramientas de software de realidad virtual mencionadas anteriormente se buscará la que tenga el mejor desempeño en las plataformas de hardware que se a continuación se listan.

[1] En la dirección: *http://vrml.sgi.com* .

[2] Los sockets son un servicio que proporciona el sistema operativo UNIX para establecer la comunicación entre procesos que pueden estar corriendo en la misma máquina o en máquinas diferentes.

- Una estación de trabajo modelo O2 (Figura 4.2).

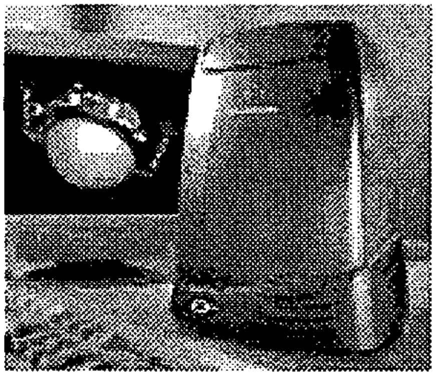


Figura 4.2. Computadora Silicon Graphics modelo O2.

- La supercomputadora CRAY-ORIGIN 2000 (Figura 4.3).

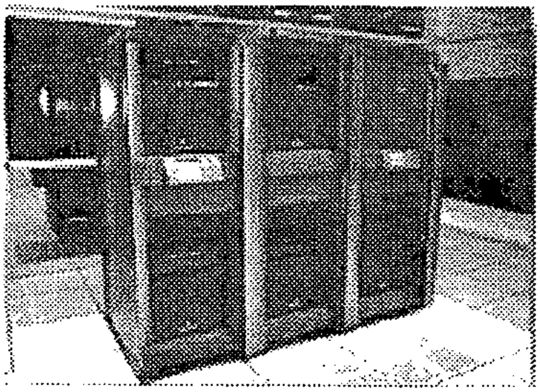


Figura 4.3. Supercomputadora CRAY-ORIGIN 2000.

Como comentario adicional, cabe mencionar que la estación de trabajo O2 cuenta con un equipo de realidad virtual. Éste se compone de un casco de realidad virtual (HMD), un Mouse 3-D (o joystick), un guante (Data Glove) y una unidad de procesamiento de entradas (IPU). En la figura 4.4 podemos apreciarlo.

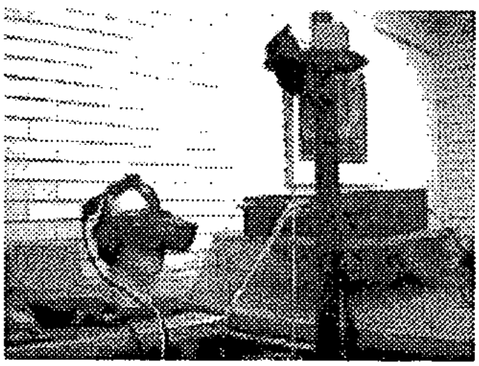


Figura 4.4. Equipo de realidad virtual del Laboratorio de Interfaces Inteligentes.

A manera de conclusión, cabe señalar que dentro de todas las herramientas y equipos de realidad virtual que se encuentran disponibles en el mercado, se eligieron las anteriores por que cubrían las necesidades que requería este proyecto, además de ser las más utilizadas de acuerdo a las listas de discusión de usuarios de realidad virtual y a las opiniones de la gente que asiste cada año a la convención mundial SIGGRAPH. Con respecto al hardware, se eligió la plataforma Silicon Graphics debido a que está orientada al trabajo con gráficos y animaciones por computadora, proporcionando un vasto soporte en hardware para este tipo de aplicaciones.

Capítulo 5.

Caso de estudio: Construcción del espacio virtual del Laboratorio de Interfaces Inteligentes.

5.1 Antecedentes.

En 1997 el Laboratorio de Interfaces Inteligentes de la Facultad de Ingeniería a cargo del Dr. Jesús Savage Carmona, adquirió un equipo de realidad virtual, el cual incluía una estación de trabajo Silicon Graphics O2, un casco (Head-Mounted Display), un Mouse 3-D (o joystick), un guante de datos (Data Glove) y el paquete dVISE de la compañía DVISION.

El proyecto en el cual se utilizaría este equipo, consistía en la creación de espacios virtuales en los que se pudieran manipular robots, es decir, se tomaría un espacio real y se construiría virtualmente. Cuando el usuario modificara la posición del robot "virtual", el robot "real" debería de moverse hacia la nueva posición en la que había quedado el robot "virtual". El robot "real" sería entonces manipulado a distancia, a través de Internet. A este tipo de aplicaciones se le conoce como telepresencia [DES97].

Antes de iniciar con un proyecto de tal magnitud, era necesario conocer que otras alternativas de software existían en el mercado, además de investigar si la estación de trabajo O2 podría correr una aplicación de tal magnitud en forma óptima, ya que la cantidad de datos a procesar era demasiado grande, además se debía transmitir información acerca de la posición del robot por Internet.

En abril de ese mismo año llegó a la U.N.A.M. la supercomputadora CRAY-ORIGIN 2000 de Silicon Graphics. Esta máquina cuenta con 40 procesadores MIPS R10000 (de 64 bits). Su función principal es la de apoyar a los investigadores en la realización de sus trabajos. Razón por la cual se pensó correr espacios virtuales en esta máquina.

Por políticas del Departamento de Supercómputo de la Dirección General de Servicios de Cómputo Académico de la U.N.A.M., la supercomputadora se dividió en dos máquinas, una de 8 procesadores y una de 32. La primera se utilizaría para hacer pruebas de los programas de los investigadores y establecer si éstos podían correr en la máquina de 32 procesadores. La máquina de 32 procesadores se emplearía para correr sólo los programas ya evaluados y adecuados a su arquitectura, es decir, que estuvieran paralelizados. Ésto con el fin de utilizar de forma óptima los recursos de la supercomputadora.

Antes de continuar con el proyecto era necesario evaluar que tan eficientemente corrían las aplicaciones de realidad virtual en la CRAY-ORIGIN 2000 y en la estación de trabajo O2. Se eligieron 3 herramientas de software de realidad virtual:

- dVISE de DIVISON (dvreality versión 4.0.1).
- VRML (Cosmo Player versión 1.0.1).
- Open Inventor (Programación en forma ASCII versión 2.1.3).

De ellas se seleccionaría aquella que tuviese el mejor desempeño en la ORIGIN 2000 y en la estación de trabajo O2.

Otro problema grave que se tenía (y se sigue teniendo) es la velocidad de la conexión del laboratorio a redUNAM . La conexión con la que se cuenta actualmente es muy lenta y las aplicaciones distribuidas no pueden correr de forma óptima.

El problema consistía entonces en:

- Crear un espacio virtual.
- Hacer pruebas de ejecución en diferentes plataformas de software y hardware.
- Elegir el producto que tuviese el mejor desempeño.

Se decidió que las pruebas de ejecución consistirían en una caminata a través de un espacio virtual. El espacio virtual sería la representación virtual del Laboratorio de Interfaces Inteligentes. El recorrido se haría utilizando dVISE, VRML y Open Inventor en la estación de trabajo O2 y en la supercomputadora ORIGIN 2000. Los resultados servirían como parámetros de selección.

5.1.1 Principios de procesamiento distribuido.

Una vez seleccionado el producto de mejor desempeño se correrán los espacios virtuales en forma distribuida, es decir, el proceso fuerte se hará en la supercomputadora y el despliegue se enviará a la estación de trabajo O2, la cual sólo estará encargada de obtener las entradas de los dispositivos conectados a ella. A este tipo de ejecución se le denomina colaborativa o distribuida [TAN93], ya que el control completo de la aplicación no lo tiene ninguna de las dos máquinas, se podría decir que las dos se auxilian entre sí para correr el software [TAN93]. Cabe señalar que las pruebas de ejecución de nuestro caso de estudio no consideran la ejecución colaborativa, ya que sale del alcance de este trabajo.

Para poder tener una ejecución distribuida es necesario que las máquinas se comuniquen y sincronicen sus procesos. El sistema operativo UNIX proporciona varias herramientas que nos ayudan a realizar esta tarea. Los sockets son una de ellas. Los sockets son funciones que permiten la comunicación entre dos procesos, los cuales pueden estar corriendo en la misma máquina o en máquinas diferentes. Cuando dos procesos se encuentran corriendo en dos computadoras diferentes y desean comunicarse por medio de sockets, utilizan la interfaz de red [MAR96].

Por otra parte se debe tener un área de memoria en la cual se puedan compartir datos, ésto con el fin de trabajar de manera conjunta, a estas áreas de memoria se les llama de diferente forma, sin embargo la mayoría de los autores coincide en llamarlas arenas. En las arenas, las secciones críticas[1] son protegidas por medio de los semáforos[2] que proporciona el sistema operativo, con lo que se garantiza la integridad de los datos [DAN97].

Al esquema mencionado anteriormente se le conoce como paso de mensajes. Es importante señalar que este tipo de aplicaciones son muy vulnerables al ancho de banda del medio físico de transmisión, por lo que se recomienda utilizarlas en redes locales de poco tráfico, en redes de alta velocidad o a horas en las que la mayoría usuarios no estén utilizando la red.

Para programar aplicaciones de este tipo nos podemos apoyar en bibliotecas de funciones creadas especialmente para trabajar de forma distribuida. Una de las bibliotecas más utilizada es PVM (Parallel Virtual Machine). Esta biblioteca necesita estar instalada en cada una de las máquinas en las cuales se desea procesar información. Sin embargo sigue siendo muy dependiente del tráfico y de la velocidad de la red en la cual estemos corriendo nuestros programas [SIL96].

A manera de conclusión, cabe señalar que existen varios algoritmos para realizar trabajo de forma distribuida, sin embargo lo único que varía de uno a otro es la forma en cómo transmiten los datos, por lo que siguen teniendo como cuello de botella el tráfico de la red [GOL93]..

5.1.2 Análisis de velocidad de la conexión del Laboratorio de Interfaces Inteligentes a RED-UNAM.

Como vimos en la sección anterior, la velocidad de transmisión de la red es uno de los factores que más afectan la ejecución de un programa cuando éste corre de forma distribuida. Por esta razón se pensó hacer un estudio la velocidad de transmisión que se logra por medio de RED-UNAM, desde la Supercomputadora CRAY-ORIGIN 2000 hasta la estación de trabajo O2 del Laboratorio de Interfaces Inteligentes.

Comercialmente existen productos que determinan cuál es la velocidad de transmisión de una red, sin embargo además de que son caros, sólo introducen más tráfico en nuestra red, por lo que la medida que nos dan es poco confiable. De forma gratuita sólo existen "sniffers"[3] que cuando más nos proporcionan porcentajes del tipo de paquetes que circulan en nuestra red. Estos programas son difíciles de compilar y poner en operación además de ser peligrosos para la seguridad de nuestra máquina y de la red en general. En ocasiones pueden hacer que la máquina en la que están corriendo se sature y se "caiga", es decir, deje de trabajar.

---

[1] Una sección crítica es un modificación o acceso a la memoria compartida.

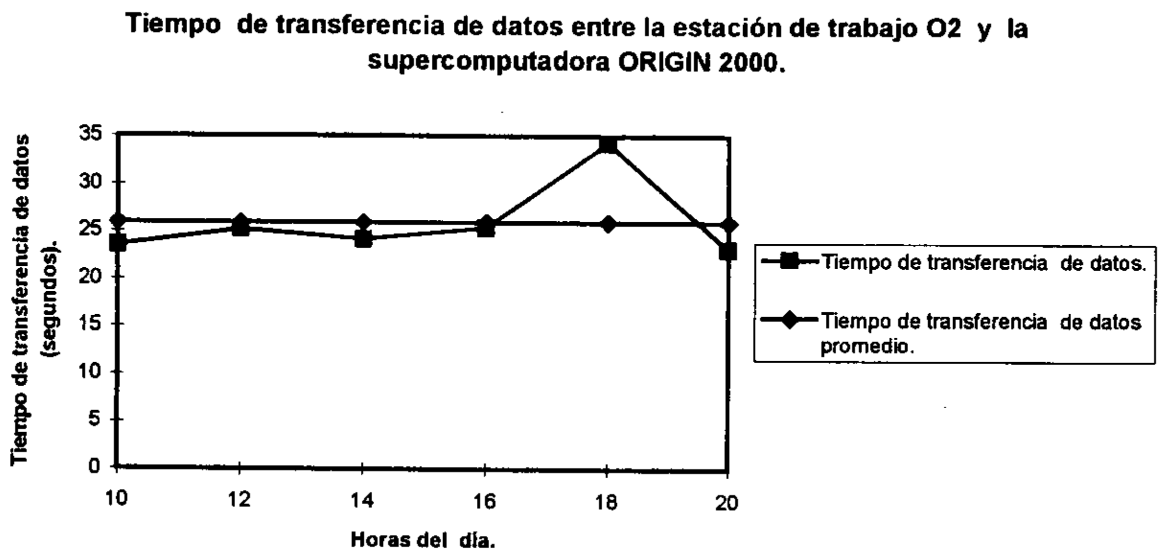
[2] Un semáforo es una variable que nos indica si un recurso de la máquina se encuentra disponible o no.

[3] Un sniffer es un programa que atrapa todos los paquetes que viajan por la red y en algunos casos los graba a disco.

Por esta razón, para poder medir la velocidad de la red se utilizó el comando *ftp(1)*, este comando sirve para transferir archivos de una computadora a otra. Cuando acaba de transferir el archivo, *ftp* nos indica la cantidad de bytes que transfirió, el tiempo que empleó y la velocidad en Kbytes/segundo a la que lo hizo. A continuación se muestra una corrida típica del comando *ftp*.

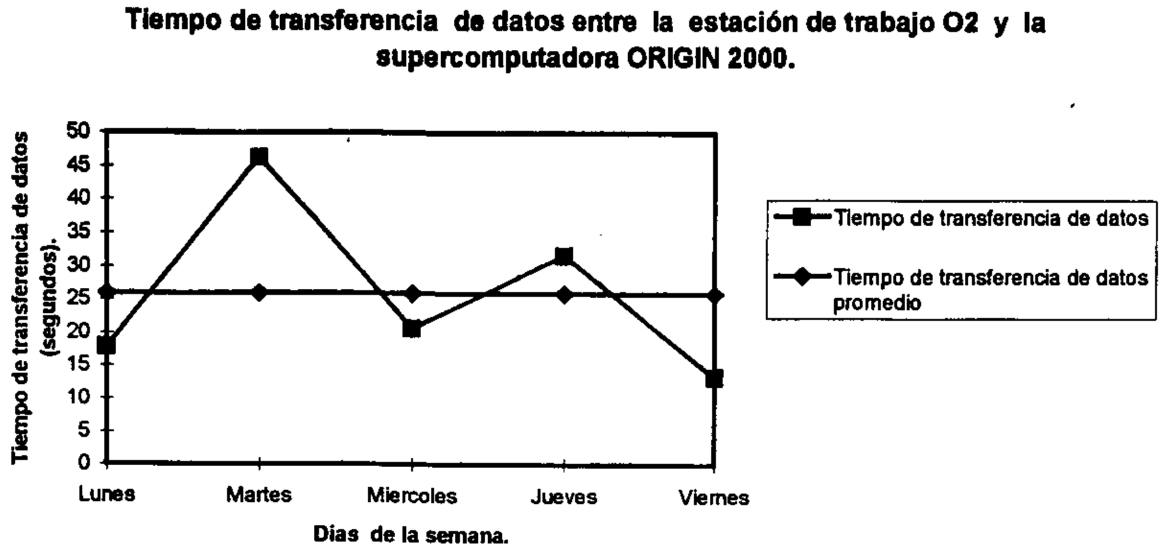
```
ftp> get mxorigin2000.jpg
local: mxorigin2000.jpg remote: mxorigin2000.jpg
200 PORT command successful.
150 Opening BINARY mode data connection for 'mxorigin2000.jpg' (13116 bytes).
226 Transfer complete.
13116 bytes received in 0.23 seconds (56.48 Kbytes/s)
ftp>
```

Para poder tener una medida representativa del tiempo de transferencia de datos, se empleó un archivo de 10 Megabytes. Éste fue transferido de la supercomputadora a la estación de trabajo O2 y viceversa, a diferentes horas del día, durante cinco días hábiles, del 12 de enero de 1998 al 16 de enero de 1998. A continuación se muestran las gráficas de los resultados obtenidos, gráfica 5.1,5.2,5.3 y 5.4.



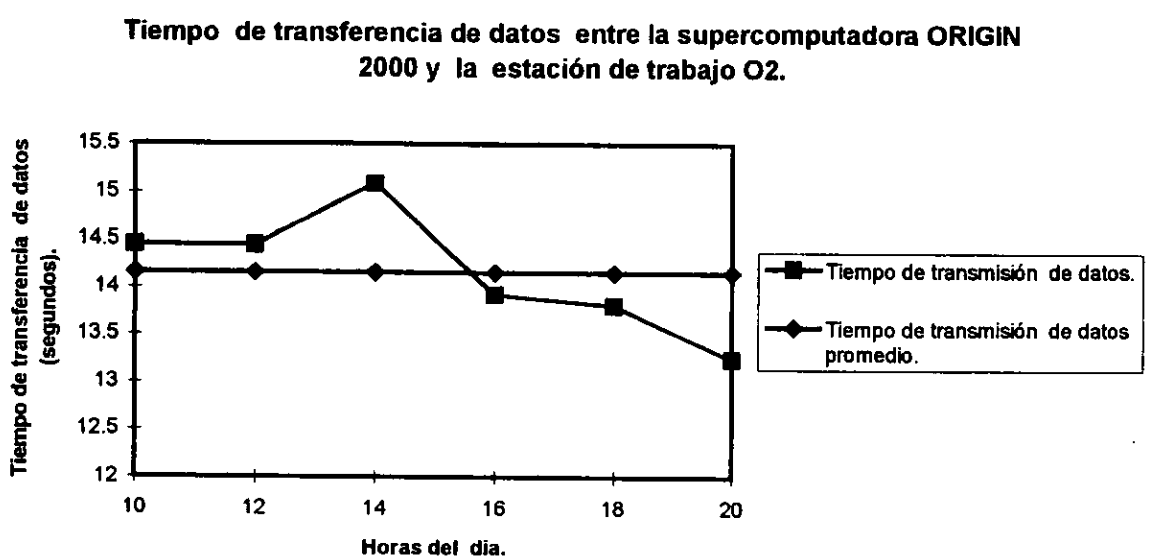
Gráfica 5.1. Tiempo de transferencia de datos entre la estación de trabajo O2 y la supercomputadora ORIGIN 2000 durante el día.

De la gráfica 5.1 se observa que el tiempo promedio de transmisión de datos de la estación de trabajo a la supercomputadora durante un día fue de: 25.91 segundos para un archivo de 10 Megabytes. Podemos observar que el tiempo se incrementa durante la tarde y disminuye en la noche, posiblemente se deba a que la red de la división de ingeniería eléctrica es muy utilizada en las tardes por los alumnos de la Facultad de Ingeniería.



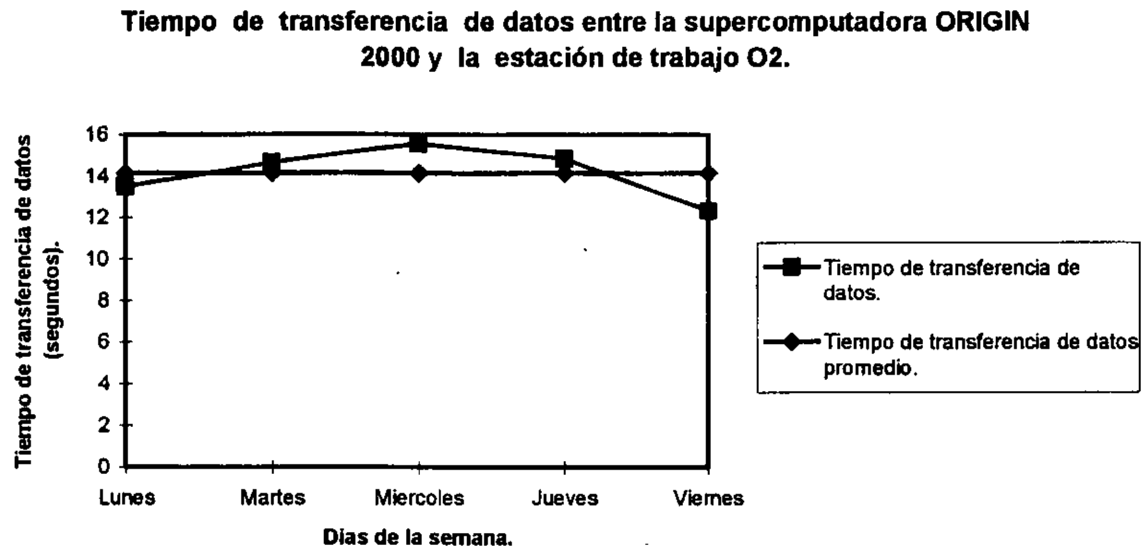
Gráfica 5.2. Tiempo de transferencia de datos entre la estación de trabajo O2 y la supercomputadora ORIGIN 2000 durante la semana.

Como podemos ver en la gráfica 5.2, el tiempo promedio de transmisión de datos de la estación de trabajo a la supercomputadora durante una semana fue de: 26.12 segundos para un archivo de 10 Megabytes. Podemos observar que la transferencia de datos se hace más lenta a mitad de la semana, mientras que al inicio o al final de ésta el tiempo disminuye de forma considerable.



Gráfica 5.3. Tiempo de transferencia de datos entre la supercomputadora ORIGIN 2000 y la estación de trabajo O2 durante el día.

La gráfica 5.3 nos muestra que el tiempo promedio de transmisión de datos de la supercomputadora a la estación de trabajo durante el día es de: 14.15 segundos para un archivo de 10 Megabytes. Podemos apreciar como el tiempo baja a medida que el día avanza. Ésto se debe posiblemente a que la ORIGIN 2000 ejecuta procesos que se encuentran en colas de trabajo durante la noche y parte de la mañana.



Gráfica 5.4. Tiempo de transferencia de datos entre la supercomputadora ORIGIN 2000 y la estación de trabajo O2 durante la semana.

Finalmente en la gráfica 5.4 podemos apreciar que el tiempo promedio de transmisión de datos de la supercomputadora a la estación de trabajo durante la semana es de: 14.61 segundos para un archivo de 10 Megabytes; por otra parte al igual que en la gráfica 5.2, la velocidad de transferencia disminuye a la mitad de la semana y se incrementa al final de ésta.

En las gráficas anteriores se puede apreciar como las velocidades de transferencia de datos son bajas, por lo que el desempeño de aplicaciones distribuidas se verá afectado notablemente. Podemos concluir entonces que, una alternativa para incrementar la velocidad de transferencia de datos, es una conexión directa entre la supercomputadora ORIGIN 2000 y la estación de trabajo O2, o en su defecto una conexión directa al backbone[1] de redUNAM.

5.2 Propuesta de solución.

Una vez identificado y delimitado el problema, se procedió a dar solución a cada una sus partes:

[1] El backbone es una red de alta velocidad que permite comunicar varias redes de computadoras, generalmente redes de área local.

- Construcción de un espacio virtual.

- Pruebas de ejecución en diferentes plataformas de software y hardware.

- Elegir el producto que tuviese el mejor desempeño.

Para la construcción del espacio virtual, se decidió modelarlo en el paquete de Silicon Graphics, Alias|Wavefront. La creación de materiales y el mapeo de texturas se harían con el paquete para PC 3D-Studio.

La razón para usar 3d-Studio, fue que se tenían a disposición los convertidores de formato para Open Inventor, VRML y dVISE.

Una vez terminado el modelo y aplicados los convertidores, se tenían tres diferentes versiones del mismo espacio virtual, una para Open Inventor, otra para VRML y una para dVISE.

Las pruebas de ejecución para el software, consistirían en el recorrido del espacio virtual. Las pruebas en la supercomputadora se realizarían utilizando una de las máquinas del Laboratorio de Visualización (del departamento de supercómputo, D.G.S.C.A.) como servidor de despliegue. Lo anterior debido a que la supercomputadora no tiene consola, por lo que la única forma de trabajar en ella es mediante sesiones remotas. Las máquinas del laboratorio de visualización se encuentran conectadas por medio de un anillo de fibra óptica, al que también están conectadas las dos supercomputadoras. Por lo que no se tienen problemas de velocidad con la red, pudiéndose realizar las pruebas de forma confiable.

Finalmente la elección del producto de mejor desempeño, se haría mediante la información proporcionada por la salida de los comandos *timex(1) y perfex(1).*

5.3 Construcción del espacio virtual.

Como se mencionó en el apartado anterior, el modelado del espacio virtual se hizo en el paquete Alias|Wavefront. Este paquete tiene una amplia variedad de herramientas que facilitan esta labor, mediante el uso de una interfaz gráfica. Por otra parte ofrece todas las ventajas del sistema operativo UNIX.

Este paquete está corriendo en una supercomputadora[1] Silicon Graphics modelo Onyx Real Engine 2. Esta máquina poseé seis procesadores MIPS R10000 a 195 MHz., y se encuentra en el Laboratorio de Visualización del departamento de supercómputo de DGSCA.

La razón para utilizar Alias en lugar de un paquete para PC como AutoCAD fue que estando el modelo en plataforma Silicon Graphics, podríamos observar que tan pesado era su procesamiento y con ésto variar el número de polígonos del modelo hasta que tuviésemos un modelo los más representativo posible. La figura 5.1 muestra la interfaz gráfica de Alias.

[1] Aunque propiamente esta máquina no puede clasificarse como una supercomputadora, la gente de Silicon Graphics la considera como tal.

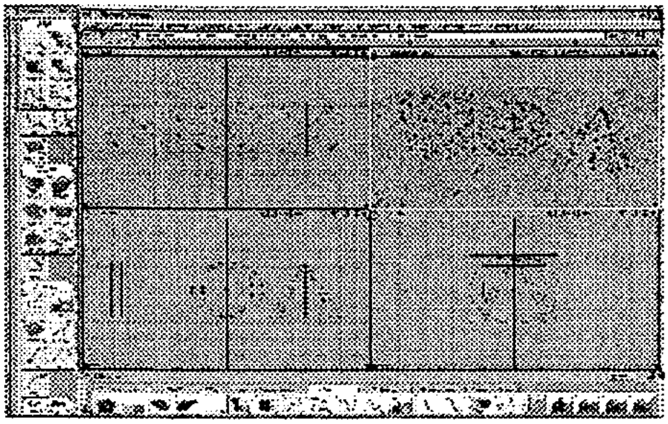


Figura 5.1. Interfaz gráfica de Alias|wavefront.

Una vez terminado el modelo, se exportó al formato DXF, que es compatible con 3D-Studio. La figura 5.2 muestra el modelo alambrado del espacio virtual. Posteriormente se definieron las texturas y los materiales de los objetos del espacio virtual mediante el uso de 3D-Studio. La figura 5.3 muestra la una vista del espacio virtual con las texturas finales. Cabe mencionar que la definición de materiales y el mapeo de texturas es muy sencillo de realizar en 3D-Studio.

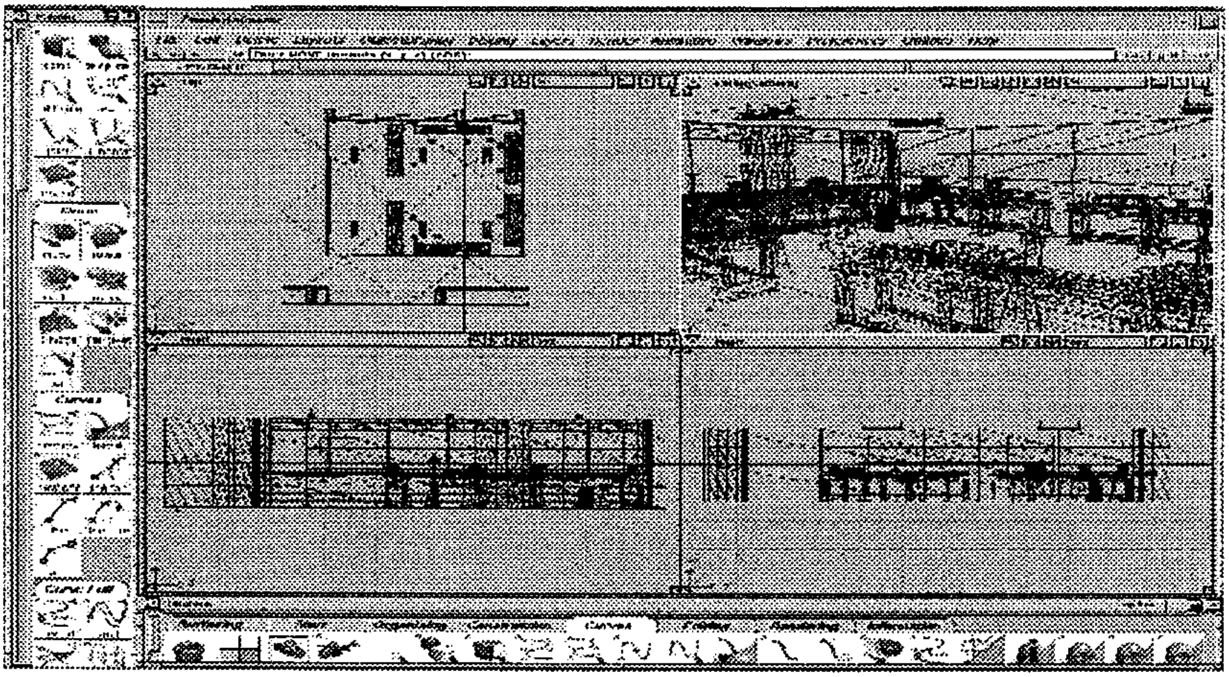


Figura 5.2. Modelo alambrado del espacio virtual.

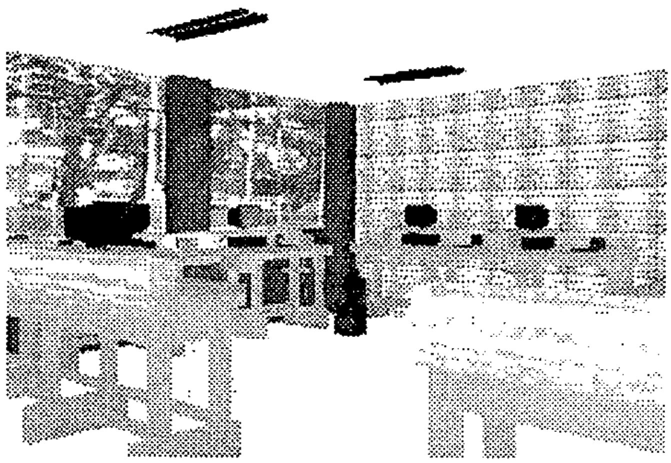


Figura 5.3. Modelo terminado.

Después de ésto se le aplicaron los convertidores de formato al modelo. Es importante señalar que en el caso de los convertidores de Open Inventor y de VRML, sólo respetan la definición de materiales, pero no el mapeo de texturas, es decir, sólo conservan los colores de los objetos e ignoran el mapeo de imágenes En el caso del convertidor de dVISE, éste conserva tanto el mapeo de texturas como los colores de los objetos, sin embargo se desactivaron las texturas para poder tener condiciones iguales para cada una de las herramientas..

Una vez construido el espacio virtual para cada una de las plataformas de software, ya se podían hacer las pruebas en la estación de trabajo O2 y en la supercomputadora ORIGIN 2000.

Capítulo 6.

Pruebas en una máquina Silicon Graphics: modelo O2.

6.1 . Características principales dė la arquitectura de una estación de trabajo O2.

Las estaciones de trabajo O2 están basadas en una arquitectura de memoria unificada (UMA[1]). Cada parte del sistema (CPU, tarjeta de video, dispositivos de I/O) tiene acceso directo al sistema principal de memoria unificada de 2.1 GB/seg., el cual sirve como un bus interconector flexible de alta velocidad. El procesador central de las estaciones de trabajo O2 puede ser un MIPS R5000 o un MIPS R10000, ambos procesadores RISC de 64 bits a 175 o 195 MHz.[SIL496].

En la arquitectura tradicional de las estaciones de trabajo (e incluso de las PC's), la memoria de video así como la de los demás dispositivos, se encuentra distribuida a través de todo el sistema. Típicamente los datos deben de ser transferidos a las diferentes tarjetas de dispositivos (por ejemplo una tarjeta PCI de gráficos o una tarjeta PCI de vídeo), las cuales tienen cada una su propia memoria local. Esta transferencia de datos es lenta y hace que se desperdicie todo el potencial del hardware, un caso claro lo tenemos en el ancho de banda del bus PCI-32, éste puede llegar a transmitir 133MB/seg., sin embargo bajo este esquema sólo se alcanza a transmitir cerca de 10MB/seg.. En la arquitectura de la O2 se elimina la necesidad de hacer una transferencia de datos de este tipo. Por ejemplo , cuando la tarjeta de gráficos termina de trabajar un bloque de datos, sólo pasa un apuntador a la tarjeta de compresión, con lo que se eliminan las copias de información y el envío de éstas a través del bus. En la figura 6.1 y 6.2 podemos ver un diagrama de la arquitectura tradicional y de la arquitectura de la O2 [SIL496].

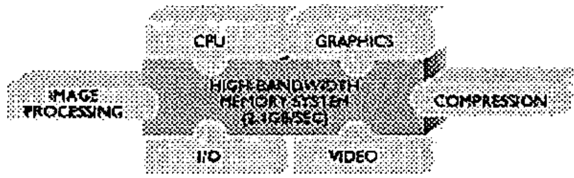


figura 6.1. Diagrama de la arquitectura de la O2.

[1] Unified Memory Architecture.

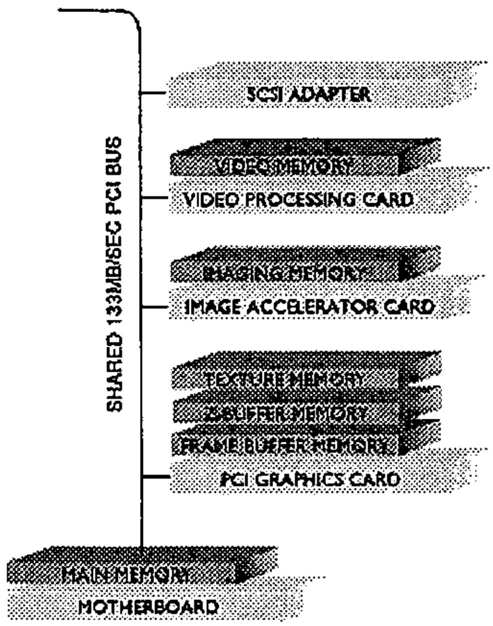


figura 6.2 . Arquitectura tradicional.

La interfaz de I/O (entrada/salida) de la O2 elimina los cuellos de botella causados por el ancho de banda del bus, ésto lo logra cubriendo sobradamente el ancho de banda de sus dispositivos, tales como la tarjeta de red, la interfaz Ultra Fast/Wide SCSI y su bus PCI de 64 bits, además de otros dispositivos estándar de entrada/salida [SIL496].

Las estaciones de trabajo O2 con procesador R10000 tienen un desempeño más alto que las estaciones que tienen un R5000, ésto debido a las unidades de punto flotante con las que cuenta el R10000, por lo que en procesos como el render es claramente visible su diferencia [SIL496].

Las estaciones de trabajo O2 cuenta con salidas de audio y video, lo cual las hace ideales para trabajar con aplicaciones gráficas, de multimedia y de realidad virtual [SIL496].

En el caso del Laboratorio de Interfaces Inteligentes la estación de trabajo O2 tiene un procesador R5000 a 175 Mhz.

A manera de conclusión, podemos decir que este tipo de computadoras son máquinas dedicadas al trabajo con gráficos, razón por la cual están optimizadas para realizar de manera eficiente procesos de este tipo.

6.2 Pruebas de ejecución del espacio virtual.

Como se mencionó en el capítulo cinco, las pruebas de ejecución consistirían en un recorrido a través del espacio virtual. La figura 6.3 nos muestra una vista superior del espacio virtual y del recorrido propuesto.

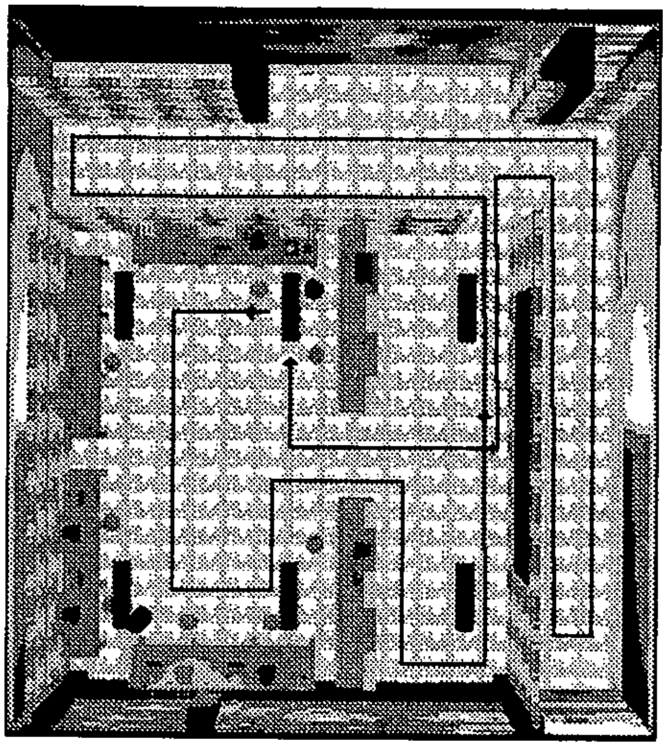


Figura 6.3 . Recorrido propuesto.

Se hicieron 30 recorridos en cada una de las herramientas de realidad virtual, teniendo un total de 90 recorridos (pruebas). Debido a que la estación de trabajo O2 tiene un procesador central R5000, sólo se pudieron obtener los tiempos de ejecución de las pruebas, ya que los comando que proporcionan información acerca de la memoria, basan sus cálculos en registros especiales que son soportados únicamente por los procesadores R8000 y R10000 de MIPS. Cabe mencionar que las pruebas se hicieron sin utilizar mapeo de texturas en el espacio virtual, únicamente se emplearon colores, ésto se debió a que los convertidores que se usaron no mapean texturas, por lo que se decidió que ninguna aplicación utilizaría esta técnica para dar igualdad de condiciones a los paquetes durante la prueba, por esta misma causa la navegación en dVISE se hizo con el Mouse 2-D y no se utilizó el casco ni el joystick (Mouse 3-D). Las figuras 6.4,6.5 y 6.6 muestran el espacio virtual corriendo en las diferentes herramientas. El espacio virtual está compuesto de 20567 polígonos. El comando utilizado para obtener el tiempo de ejecución fue *timex(1)* .

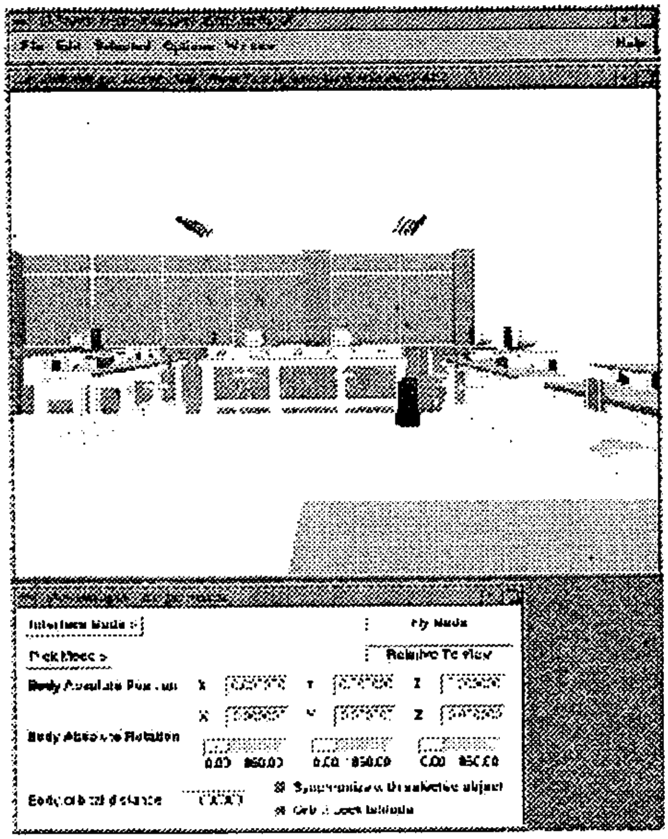


Figura 6.4 Interfaz del producto *dVISE.*

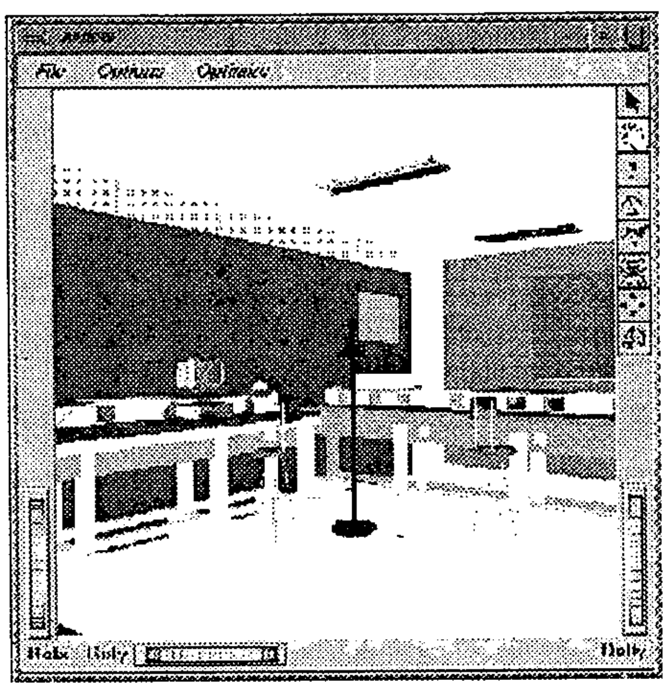


Figura 6.5 . Visor de Open Inventor *ivview.*

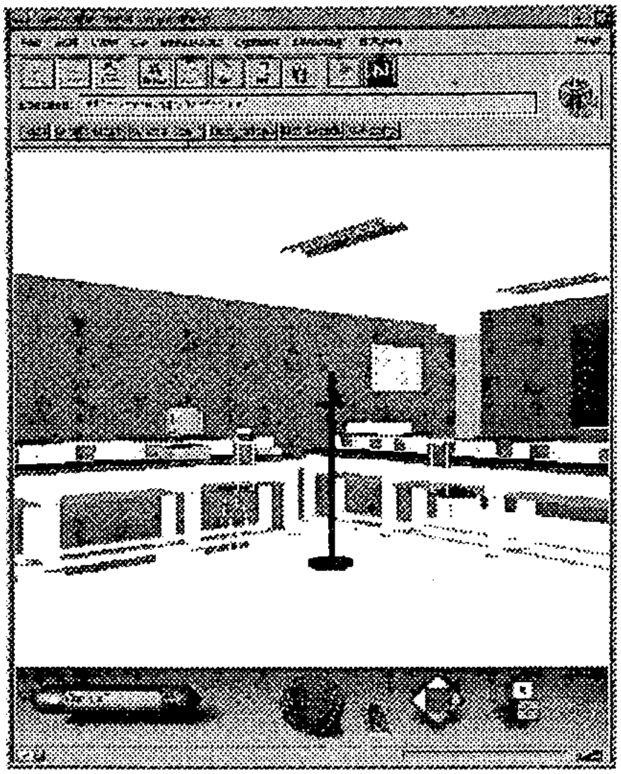
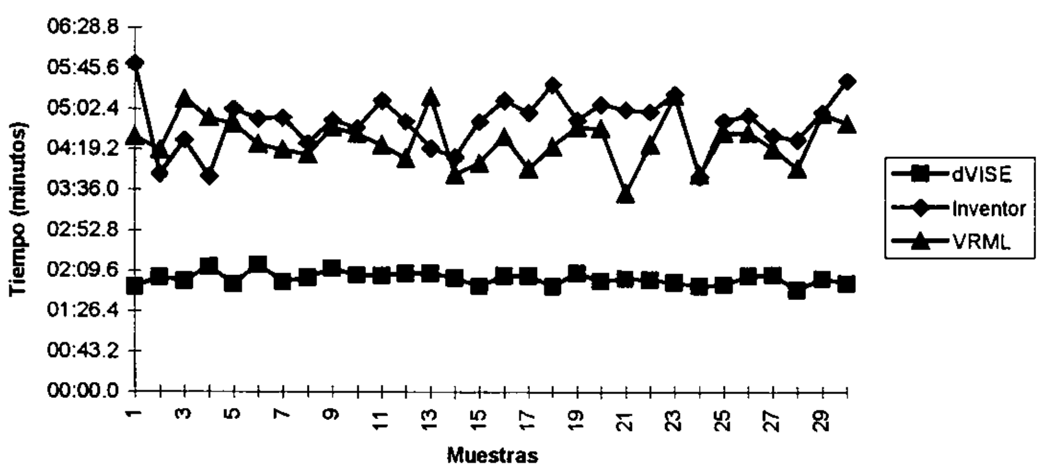


Figura 6.7. Cosmo Player . Visor de VRML bajo el navegador de Nestcape.

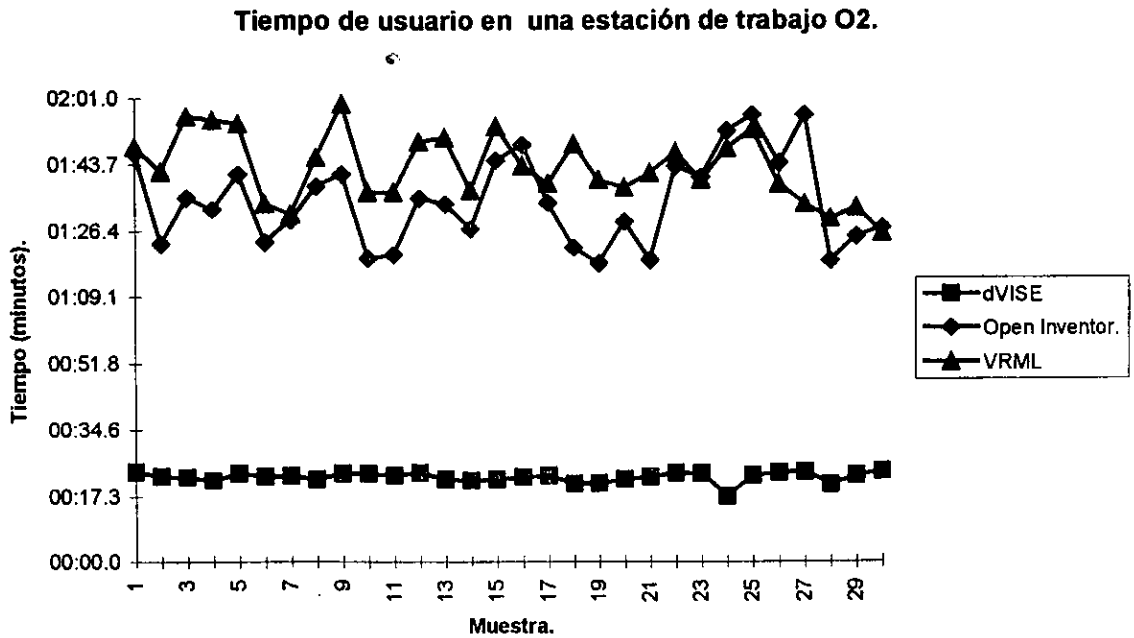
6.3 Observaciones y resultados.

A continuación se muestran las gráficas de los resultados obtenidos. Gráfica 6.1,6.2 y 6.3.

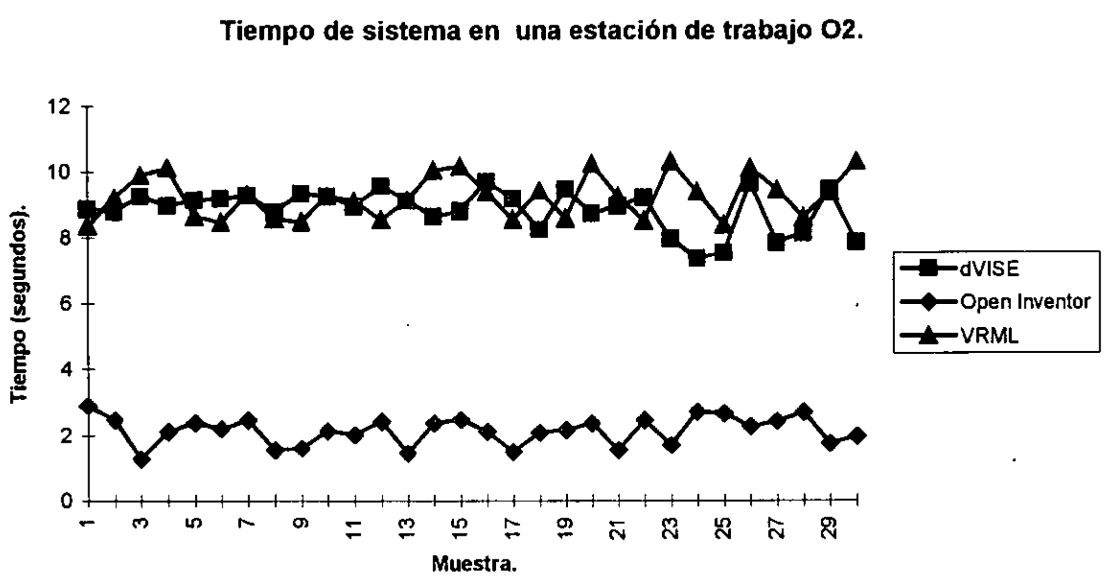
Tiempo real de ejecución en una estación de trabajo O2.



Gráfica 6.1. Tiempo real de la corrida de las herramientas de software de realidad virtual en una estación de trabajo 02.



Gráfica 6.2. Tiempo de usuario de la corrida de las herramientas de software de realidad virtual en la estación de trabajo 02.



Gráfica 6.3. Tiempo de sistema de la corrida de las herramientas de software de realidad virtual en la estación de trabajo 02.

El comando timex proporciona tres tiempos:

- Tiempo real. Consiste en el tiempo que tarda el programa en ejecutarse, es decir, el tiempo comprendido entre la invocación del programa en la línea de comandos y el momento en que se regresa el prompt.
- Tiempo de usuario. Es el tiempo de procesador que ocupó el programa.
- Tiempo de sistema. Tiempo utilizado por el programa en llamadas al sistema[1] [MAR96].

Frecuentemente el tiempo real es más grande que la suma del tiempo de usuario y de sistema, ésto se debe a que el programa pudo haber gastado tiempo en espera de algún dispositivo o de algún dato que el usuario tenía que proporcionarle [DAN97].

Los tiempos promedios fueron:

Herramienta	Tiempo real (minutos)	Tiempo de usuario (minutos)	Tiempo de sistema (segundos)
dVISE	02:01.3	00:22.1	8.83
Open Inventor	04:48.0	01:34.0	2.13
VRML	04:28.2	01:42.7	9.23

Como podemos observar, la herramienta que menor tiempo empleó en el recorrido(tiempo real) y en el tiempo de ejecución en CPU fue dVISE. Mientras que VRML empleó menos tiempo en llamadas al sistema (tiempo de sistema).

Es interesante observar como Open Inventor y VRML no difieren mucho en el tiempo real, lo cual significa que emplean casi el mismo tiempo en hacer un recorrido. También podemos observar que VRML utiliza menos tiempo en llamadas al sistema, por lo que podemos deducir que emplea de forma óptima los recursos de la máquina.

Finalmente es importante señalar que la mayor parte del tiempo empleado en la ejecución de cada una de las herramientas, se gasta esperando a que el usuario dé alguna entrada (por ejemplo un click[2]), por lo que se puede deducir que los programas que mejor emplean el tiempo de procesamiento son los programas que no interactuan con el usuario.

---

[1] La llamadas al sistema son funciones que proporciona el Kernel de UNIX.

[2] Se entiende por click el presionar cualquiera de los botones del mouse.

Capítulo 7.

Pruebas en una máquina Silicon Graphics: modelo CRAY-ORIGIN 2000

7.1 Principios básicos de paralelismo.

A principios de la década de los 1970's, comenzaron a aparecer computadoras que tenían varios procesadores operando en paralelo y computadoras cuyo hardware tenía instrucciones para realizar operaciones con vectores. A las últimas se les conoce como computadoras vectoriales y a las primeras se les conoce como computadoras paralelas.

Las computadoras vectoriales utilizan una técnica llamada pipelining, la cual consiste en la segmentación de una unidad aritmética en diferentes partes, cada una con capacidad de ejecutar una subfunción con un par de operandos. En la figura 7.1 podemos ver un ejemplo.

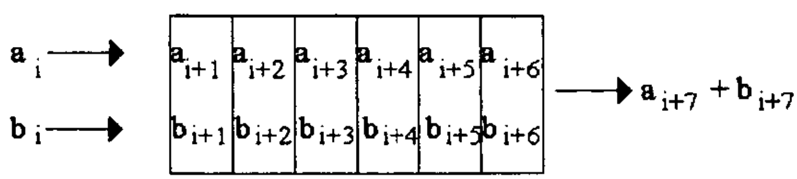


Figura 7.1 .Unidad aritmética de punto flotante con *Pipeline*.

En la figura 7.1, se muestra una unidad aritmética de punto flotante segmentada en seis secciones, cada una hace una parte de la suma. Cada segmento trabaja con un par de operandos, por lo que seis pares de operandos pueden estar en el pipeline en un tiempo dado. La ventaja de esta segmentación es que la operación es ejecutada 6 veces más rápido (o en general, K veces más rápido, donde K es el número de segmentos) que una unidad aritmética que acepta un par de operandos y computa el resultado antes de aceptar el siguiente par de operandos. Sin embargo para poder utilizar este tipo de procesamiento, los datos deben de cargarse rápidamente para tener siempre lleno el *pipeline* . Generalmente cada suboperación de las mostradas en la figura 7.1 son hechas cada ciclo de reloj. Nuevos datos estarán listos para entrar al *pipeline* cada ciclo . Algunas máquinas tienen instrucciones de hardware para operar vectores, con lo que se elimina la necesidad de ejecutar de forma separada las instrucciones de carga y almacenamiento de datos; con una instrucción se controla la carga de los operandos y el almacenamiento de los resultados. Una de las primeras computadoras vectoriales fue la CDC STAR-100 construida por Control Data Corporation en 1973.

Las computadoras Cray, al igual que muchas otras, tienen unidades de pipeline para suma y multiplicación. Algunas computadoras vectoriales ofrecen además unidades aritméticas separadas para aritmética escalar. Estas unidades también utilizan el pipeline, pero no aceptan operaciones con vectores [GOL93].

Analicemos ahora las computadoras paralelas. La idea básica de una computadora paralela es que cierto número de procesadores trabaje en forma cooperativa para ejecutar una sóla tarea. Idealmente, si a un procesador le toma un tiempo t hacer una tarea, entonces p procesadores harán la misma tarea en un tiempo de t/p. Sin embargo el ejemplo anterior es un caso perfecto, por esta razón la meta de los diseñadores de algoritmos es aproximarse a él tanto como sea posible. Los procesadores de una computadora paralela pueden ser vectoriales.

Existen dos grandes clasificaciones en los sistemas paralelos, éstos se dividen de acuerdo a la forma en la que van a ser controlados los procesadores, sistemas Múltiples Datos-Una Instrucción y sistemas Múltiples Datos - Múltiples Instrucciones. En un sistema de Múltiples Datos - Una Instrucción (Single-Instruction-Multiple-Data, SIMD), todos los procesadores se encuentran bajo el control de un procesador central, llamado controlador, el resto de los procesadores se encuentran haciendo todos la misma instrucción (o nada) en un tiempo dado. La IlliacIV, fue la primera máquina masivamente paralela, (se construyó a principios de los 1970's por Thinking Machines, Inc.) esta computadora era una SIMD. Otra máquina la CM-2 era una máquina SIMD con 64936 procesadores de un bit.

En las máquinas de Múltiples Datos-Múltiples Instrucciones, cada procesador corre bajo su propio control, ésto permite gran flexibilidad en la ejecución de tareas que un procesador puede hacer en un tiempo dado. Pero también introduce el problema de la sincronización. Para muchos problemas, los programas en cada procesador puede ser idénticos (o muy parecidos). Por lo que todos los programas estarían haciendo las mismas operaciones en diferentes conjuntos de datos, de igual forma en que lo haría una computadora SIMD. Este problema dió origen al modelo computacional Múltiples Datos-Solo un Programa, también conocido como modelo de datos paralelos.

Nuevamente volvemos a tener dos clasificaciones dentro del modelo de datos paralelos, el modelo de memoria compartida y el modelo de memoria distribuida. Un sistema de memoria compartida se muestra en la figura 7.2. Aquí todos los procesadores tienen acceso común a la memoria. Cada procesador puede también tener su propia memoria local para código de programa y resultados intermedios. La memoria compartida podría ser usada entonces para datos y resultados que son utilizados por más de un procesador. Todas las comunicaciones entre procesadores se hacen a través de la memoria compartida. Una gran ventaja de los sistemas de memoria compartida es la velocidad de transmisión de datos entre procesadores. Por otro lado, una de sus más serias desventajas es que diferentes procesadores pueden querer usar una misma área de memoria, en este caso existirá un retraso de tiempo hasta que la memoria esté de nuevo libre. Este retardo es llamado *tiempo de conexión* y puede crecer de acuerdo al incremento en el número de procesadores del sistema.

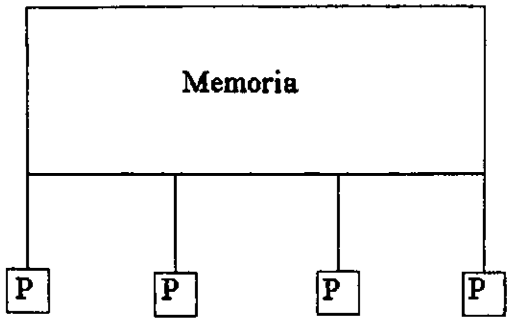


Figura 7.2. Sistema de Memoria compartida.

Una alternativa a los sistemas de memoria compartida son los sistemas de memoria distribuida, en los cuales cada procesador puede accesar a su propia memoria local. La comunicación entre procesadores se lleva a cabo mediante el *paso de mensajes*, el cual consiste en la transferencia de datos u otra información entre procesadores [GOL93].

Un aspecto importante e interesante de las computadoras paralelas es como se van a comunicar los procesadores entre sí. Ésto también es importante para los sistemas de memoria compartida.

Existen varios esquemas de comunicación, entre los más comunes podemos encontrar:

- Conexión completa. En este tipo de enlace cada procesador tiene una conexión directa con los demás procesadores. Ésto es teóricamente el esquema ideal de conexiones, sin embargo es poco práctico para un número muy grande p de procesadores, ya que se requerirían $p - 1$ líneas para cada procesador.

- Switches. Una aproximación de un sistema de conexión completa se obtiene por medio de switches cruzados, los cuales podemos conectar a cada memoria . Este esquema tiene la ventaja de permitirle a cualquier procesador el acceso a cualquier memoria con un número pequeño de líneas de conexión. La figura 7.3 nos muestra este esquema.

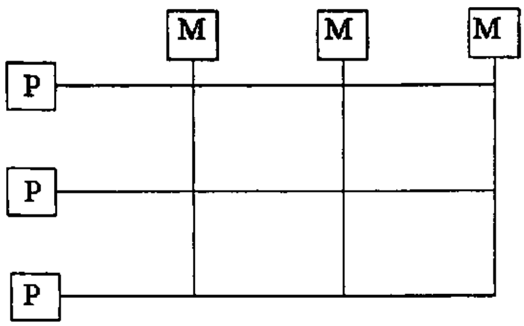


Figura 7.3. Esquema de switches cruzados.

Una desventaja de los switches cruzados es que se requieren p^2 switches para conectar p procesadores a p memorias. Ésto es poco práctico para un número grande p, pero puede ser mejorado con una red de switches como la mostrada en la figura 7.4 . En esta red, ocho procesadores están conectados a ocho memorias. Cada caja representa un switch de dos caminos y las líneas son ligas de transmisión. Por medio de esta topología cualquier procesador se puede conectar con cualquier memoria.

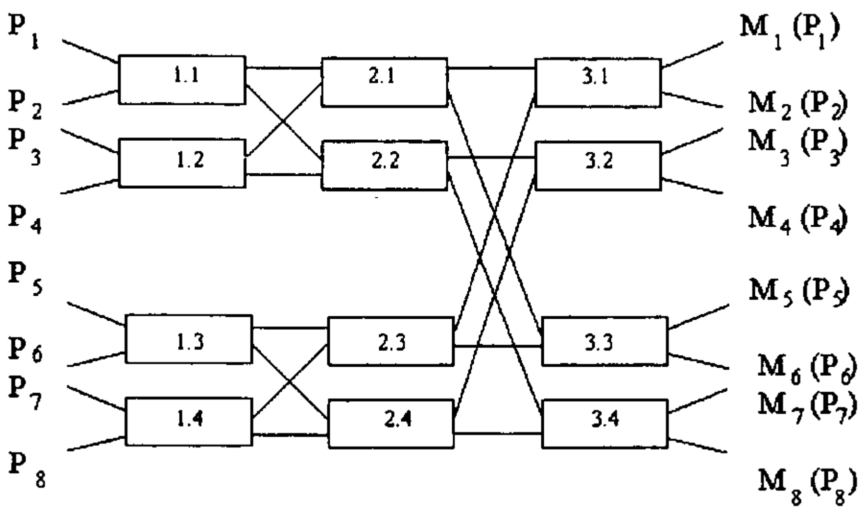


Figura 7.4. Esquema de red de switches.

Bajo este esquema sólo se necesitan $p/2$ switches para cada capa de switches y $log_2\ p$ capas, lo que nos da un total de $\frac{1}{2}\ p\ log_2 p$ switches.

- Conexión en malla. Uno de los más populares esquemas de intercomunicación es conectar cada procesador con solo algunos de los procesadores vecinos. En la figura 7.5 se muestra un arreglo lineal. En éste, cada procesador está conectado a los dos vecinos más cercanos. El procesador final puede estar conectado sólo a un procesador o conectado al primer procesador, en ese caso hablamos de un anillo.

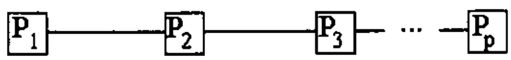


Figura 7.5. Arreglo Lineal.

Existen ciertas similitudes entre un arreglo lineal y una red de bus. En la red de bus cada procesador obtiene información del bus. El problema con la conexión de bus es que pueden haber varios procesadores que quieran enviar información por el bus al mismo tiempo. El máximo número de transmisiones que pueden ser hechas para comunicar dos procesadores del sistema es llamado *longitud de comunicación* o *diámetro* del sistema.

Muchas de las conexiones de malla existentes en la actualidad, se han construido utilizando patrones bidimensionales. Uno de los esquemas más simples de este tipo de conexión se muestra en la figura 7.6, en esta conexión cada procesador está unido mediante una cuadrícula regular, conectado con sus vecinos del norte, sur, este y oeste.

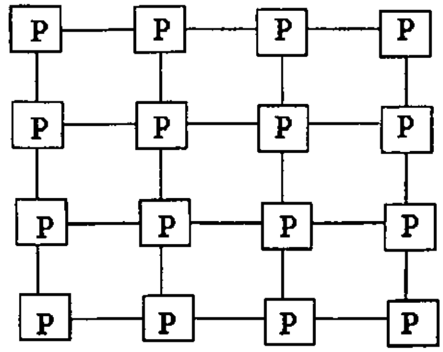


Figura 7.6. Conexión de malla.

La principal desventaja de esta topología es la transmisión de datos entre procesadores distantes, ésto debido a que los datos deben de pasar a través de una sucesión de procesadores intermediarios. Debido a que pueden existir varios caminos por donde pueda fluir la información entre dos procesadores, el primer paso en la comunicación es establecer un camino lo más directo posible. Una vez hecho ésto, el mensaje es enviado con la mínima interrupción de procesadores intermediarios. Sin embargo se siguen presentando problemas cuando dos mensajes tratan de usar al mismo tiempo la misma línea de conexión.

Conexión de hipercubo. Una variante interesante de una conexión de malla es la conexión local de procesadores en dimensiones mas grandes. Consideremos un cubo en tres dimensiones e imaginemos que un procesador se encuentra localizado en cada vértice del cubo. Las aristas del cubo serán entonces canales de comunicación entre procesadores, por lo que cada procesador estará conectado a sus tres vecinos más cercanos mediante éstas. Ahora imaginemos un esquema de conexión análoga, usando un cubo en k dimensiones. De nuevo, los procesadores estarán localizados en alguno de los 2^k vértices del cubo de dimensión k . Cada procesador está conectado a sus k vértices adyacentes, mediante las aristas del cubo; esta conexión es llamada *conexión en hipercubo* . Es obvio que no podemos construir un cubo de dimensión menor a 3, por lo que $k >= 3$.

Para un cubo de dimensión 4, hay 16 procesadores, cada uno conectado a otros 4. La figura 7.7 nos muestra una conexión para un cubo de dimensión 4. Podemos observar que un cubo de dimensión 4 puede construirse mediante 2 cubos de dimensión 3, uniendo los vértices correspondientes. En general, podemos construir un cubo de dimensión k conectando todos los vértices correspondientes de dos cubos de dimensión $k - 1$.

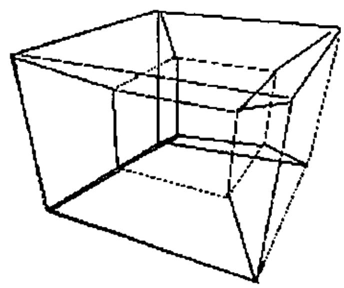


*Figura 7.7.*Conexión de un cubo de dimensión 4.

En un esquema de conexión de hipercubo, el número de conexiones para cada procesador se incrementa conforme al número de procesadores del sistema en una proporción de $log_2\ p$. Por ejemplo para una máquina de 64 procesadores (un cubo de dimensión 6), cada procesador es conectado a otros seis procesadores y para un sistema de 1024 procesadores, cada procesador se conecta a diez diferentes procesadores.

- Clusters. Un esquema de cluster se muestra en la figura 7.8. Aquí tenemos n clusters, cada uno compuesto de m procesadores. En cada cluster los procesadores pueden estar conectados en alguno de los esquemas descritos anteriormente. Lo mismo puede hacerse con los clusters, es decir,se pueden conectar en cualquier tipo de esquema. La comunicación en cada cluster se conoce como local, mientras que la comunicación entre clusters es llamada global. En este esquema se busca tener un balance entre éstos dos tipos de comunicación. Idealmente la comunicación local debería de ser más frecuente que la comunicación global. Los cluster a su vez pueden agruparse para formar clusters de clusters y así sucesivamente [GOL93].

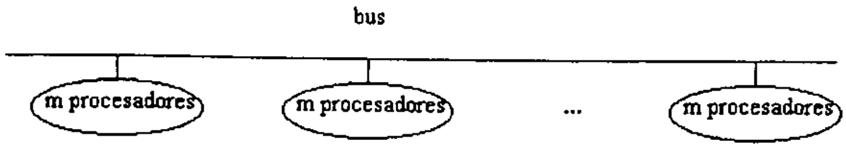


Figura 7.8. Conexión de cluster.

A continuación hablaremos acerca de la comunicación y la sincronización de tareas en un sistema de memoria distribuida.

Consideremos el problema típico de enviar n números de punto flotante de la memoria de un procesador P_1, a la memoria de otro procesador P_2. Primero los datos deben de ser cargados a un buffer de memoria en P_1. Despues deben de enviarse mediante un comando que transferirá los datos a un buffer de memoria de P_2. Finalmente P_2 ejecutará un comando para recibir los datos que serán enviados a su destino final en su memoria. En este tipo de operaciones pueden ser utilizados Co-procesadores para aligerar la carga de trabajo del procesador central.

El tiempo que toma realizar esta operación, varia de máquina a máquina, pero en general, si se realizan frecuentemente, el rendimiento de la máquina puede caer considerablemente. Una forma de solucionar este problema, es particionar la memoria en páginas, las cuales consisten en bloques contiguos de localidades de memoria. Por lo que la dirección de una palabra de memoria podría consistir en un número que nos indicara la página y un segundo número que nos indicara la localización del dato dentro de la página. Si una palabra de memoria ha sido leída recientemente de una página de memoria, una subsecuente lectura de la misma página es llamada *lectura cercana* y no representa ningún retraso para el sistema. Si por el contrario una página diferente es leída, se llama *lectura lejana* y representa un retardo de 10 o más ciclos de reloj. Un ejemplo lo podemos ver en la suma de vectores:

$$z_i = x_i + y_i \qquad i = 1, \dots, n.$$

Supongamos que n=5000 y que los vectores x y y son almacenados en localidades contiguas de memoria. Para un pagina de 4096 bytes de tamaño, por cada operación se tienen que hacer 2 lecturas a diferentes páginas, ésto mismo ocurre con la escritura. Por lo que el tiempo para cada suma se incrementa de 35 a 40 ciclos más. Como podemos observar se obtendría un mejor desempeño si los vectores fueran almacenados de forma entrelazada $x_1, y_1, z_1, x_2, y_2, z_2, \dots$.

Consideremos ahora el problema de sumar 2 vectores de dimisión n, a y b. Las sumas

$$a_i + b_i \qquad i=1, \dots, n,$$

son todas independientes y pueden ser hechas en paralelo. Este problema tiene un paralelismo matemáticamente perfecto. Por otro lado, no puede haber paralelismo perfecto en una computadora paralela, debido al balance de carga de trabajo, el cual consiste en la asignación de tareas a los procesadores del sistema. Se debe de mantener trabajando al procesador tanto como sea posible. En general, el balance se puede hacer estático o dinámico. En el balance estático, las tareas (y los datos para la memoria distribuida) son asignados a cada procesador al principio de un cálculo. En el balance dinámico, tareas (y datos) son asignados a cada procesador durante el transcurso del cálculo a realizar, por lo que siempre se está revisando que procesador está libre para seguirle asignando trabajo [GOL93].

El balance dinámico es más eficiente para las máquinas de memoria distribuida. Relacionado con el concepto de balance de carga, encontramos a la granularidad. Ésta se refiere al tamaño de las tareas asignadas a cada procesador. Cuando se dice que los trabajos que está realizando el sistema son de grano grueso hablamos del multiproceso de algunas aplicaciones y funciones de UNIX, tal es el caso de los procesos creados con forks[1]. El grano mediano se refiere a los programas que usan threads[2]. El grano fino consiste en la paralelización de loops, en el caso de la suma, los procesos generados por el programa serán muy pequeños, por lo que se puede tener trabajando óptimamente al sistema [DAN97].

En un sistema de memoria distribuida, los procesadores necesitarán intercambiar información varias veces durante la ejecución de un programa y la comunicación no debe interrumpir su trabajo, a ésto se le conoce como overhead.

Hay dos aspectos de la sincronización que contribuyen al overhead. El primero es el tiempo de sincronización; usualmente requiere que todos los procesadores ejecuten ciertas instrucciones. El segundo aspecto es el tiempo que algunos, o todos los procesadores están en espera de trabajo, es decir, ociosos, esperando a que los demás terminen su tarea.

Idealmente nosotros podemos ejecutar un programa *p* veces más rápido en una máquina con *p* procesadores que en una máquina con un solo procesador. Este factor es raramente alcanzado. Al factor que alcanzamos se le conoce como *speed-up* del *algoritmo paralelo*, éste se define por :

$$S_p = \frac{\text{tiempo de ejecución en un solo procesador}}{\text{tiempo de ejecución utilizando p procesadores}}$$

La eficiencia la podemos medir entonces en términos de:

$$E_p = \frac{S_p}{p}$$

Finalmente cabe mencionar que existe una ley llamada la ley de Amdhal, la cual nos permite calcular el valor de speed-up ideal, sin perdidas debidas al bus. Más específicamente, si s es la porción del programa que corre secuencialmente y p la parte que corre en paralelo (s + p =1), el speed-up paralelo es:

$$S_p = \frac{s + p}{s + p/n} = \frac{1}{s + (1 - s)/n} <= \frac{1}{s}$$

[1] Fork es una función de UNIX que permite crear una copia de un proceso.

[2] Los threads son pequeños procesos encargados de realizar una tarea común. Se utilizan principalmente para balancear la carga de trabajo del sistema

Utilizando esta formula podemos estimar las porciones secuenciales y paralelas del programa, con las que posteriormente podremos calcular el speed-up ideal para cada corrida de nuestro programa [SIL96].

7.2 Características principales de la arquitectura de la ORIGIN 2000.

Como vimos en la sección anterior, las máquinas de memoria compartida están basadas en el bus, es decir, dependen del bus para su funcionamiento. Ofrecen multitarea, pueden dar servicio a múltiples usuarios y son relativamente fáciles de programar en forma paralela, sin embargo, el ancho de banda finito del bus puede convertirse en un cuello de botella y limitar la escalabilidad[1] del sistema. Por su parte los sistemas de memoria distribuida solucionan el problema de la escalabilidad eliminando el bus, pero también se pierden las facilidades de programación de los sistemas de memoria compartida. La arquitectura de la ORIGIN 2000 se ha diseñado pensando en combinar las ventajas de ambos sistemas con su modelo escalable de memoria compartida [SIL96].

La ORIGIN 2000 utiliza físicamente memoria distribuida; sin embargo existe un bus común no demasiado grande como para convertirse en un cuello de botella. Los anchos de banda de la memoria local se incrementan al agregar más procesadores y memoria al sistema. Por otra parte el hardware de la ORIGIN 2000 trata a la memoria como si ésta estuviese unificada, ésto lo logra mediante un sistema de direccionamiento global, por lo que tenemos un sistema de memoria compartida como en los sistemas basados en el bus[SIL96].

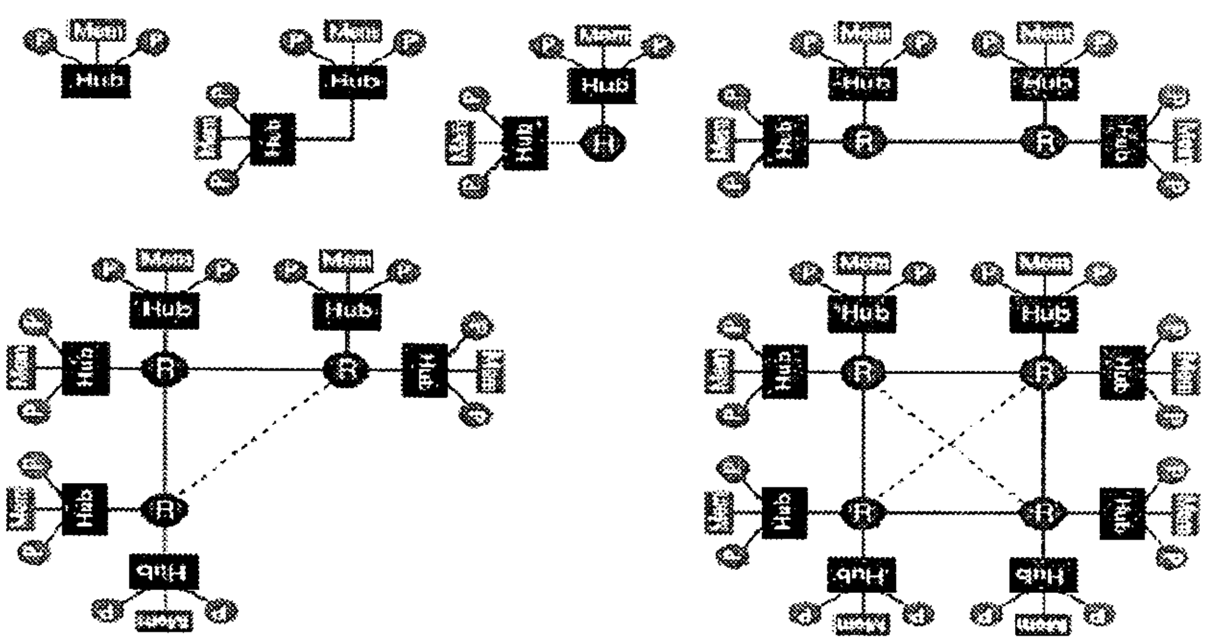


Figura 7.9. Configuraciones de sistemas ORIGIN 2000.

[1] Ecalabilidad es la capacidad de una aplicación o producto (de hardware y/o software) para continuar haciendo sus funciones tan bien como sea posible con la anexión o mejoramiento de alguno de sus componentes memoria, procesador, dispositivos periféricos, etc.

En la figura 7.9 se presentan sistemas ORIGIN que van desde 2 hasta 16 procesadores. El sistema de la esquina superior izquierda es un nodo de ORIGIN 2000. Éste consiste en uno o dos procesadores, memoria y un dispositivo llamado hub. El hub nos permite tener una arquitectura basada en el bus (de memoria compartida). El hub maneja el acceso de cada procesador a la memoria y a los dispositivos de entrada y salida. Por lo que se encarga de mandar peticiones a otros nodos en caso de requerir algún dato que no esté presente en la memoria local.

El sistema más pequeño de ORIGIN se compone de un solo nodo. Los sistemas más grandes son construidos con la conexión de múltiples nodos. En un sistema de más de 1 nodo el acceso a memoria remota es un poco más costoso que el acceso a memoria local. El hub de cada uno de los nodos, puede accesar su memoria local independientemente de los otros nodos. Un hub determina si una dirección de memoria es local o remota basándose en la dirección física del dato al que se deseé accesar [SIL96].

Cuando existen más de dos nodos en un sistema, los hubs no se pueden conectar directamente como en el caso de un sistema de dos nodos. Por lo que es necesario un dispositivo que sea utilizado por la ORIGIN para controlar el flujo de datos entre múltiples hubs, este hardware es conocido como ruteador. Cada ruteador tiene 6 puertos de comunicación, por lo que podemos conectar seis hubs en un ruteador.

Por medio de estos dispositivos podemos en teoría crear hipercubos de dimensión n. Como consecuencia de ésto no existe un límite del número de procesadores que pueden constituir efectivamente un sistema.

Para permitir la escabilidad han sido sacrificadas algunas de las características de los sistemas de memoria compartida. Por ejemplo, el tiempo de acceso a memoria no es uniforme, éste varia dependiendo de que tan lejos esté la memoria que se desea procesar. Sin embargo la electrónica de la ORIGIN 2000 uniforma éste tiempo, para lograrlo se han tomado las siguientes medidas:

- El hardware ha sido diseñado para que el tiempo de acceso a memoria remota no se muy grande.

- El R10000 opera datos que están residentes en sus cachés.

- El acceso a memoria de muchos programas se hace primero en la memoria local, es decir, si un programa solicita un dato, primero se busca en el caché primario, si no se encuentra, se busca en el caché secundario, si no está el dato se va a la memoria local de la tarjeta, si aún no se encuentra el dato , se busca en la memoria remota de algún procesador, utilizando la dirección de memoria del dato.

- El R10000 ejecuta otras instrucciones mientras espera los datos que solicitó a memoria remota.

Cada nodo de ORIGIN 2000 contiene uno o dos procesadores R10000. En el caso de Berenice[1] las tarjetas de nodo tienen 2 procesadores R10000 a 195 Mhz. con 4 MB de caché secundario. La figura 7.10 nos muestra una tarjeta de nodo.

[1] Berenice es el nombre de la supercomputadora ORIGIN 2000 de la UNAM.

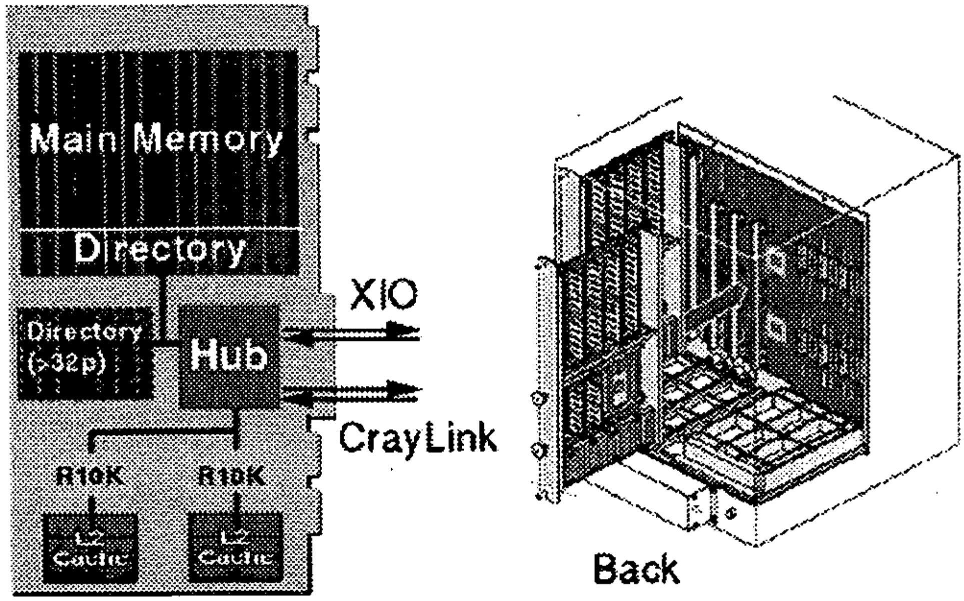


Figura 7.10. Tarjeta de nodo de ORIGIN 2000.

Cada procesador y su caché secundario están empacados en un arreglo llamado modulo de memoria horizontal en línea (HIMM). Cada R10000 tiene dos niveles jerárquicos de memoria caché. Dentro del microprocesador se encuentra el primer nivel de caché o caché primario, el cual es utilizado para instrucciones y tiene un tamaño de 32 KB. El caché secundario es utilizado para datos e instrucciones, su tamaño puede ir de 512 KB a 16 MB; el tamaño del caché secundario en los nodos de ORIGIN 2000 a 195 Mhz. es de 4MB. Además de ésto, las tarjetas de nodo tienen 16 sockets para memoria. Los módulos de memoria son instalados en pares, cada uno de los cuales recibe el nombre de Módulos Duales de Memoria en Línea (DIMM), cada DIMM representa un banco de memoria. Ocho pares de DIMMs pueden ser instalados en una tarjeta de nodo.

Cada DIMM viene en tres tamaños.

- 64 MB
- 128 MB
- 512 MB

Por lo que cada tarjeta de nodo puede tener de 64 MB a 4GB de memoria. Además de la memoria especificada en el tamaño del DIMM, éste contiene memoria extra, parte de la cual es usada para detección y corrección de errores. El resto de esta memoria es conocida como *directorio*. Dado que no existe un bus común, la ORIGIN tiene un mecanismo llamado coherencia de caché basado en el directorio, el cual sirve para mantener la integridad de los datos en todo el sistema [SIL96].

El componente final de la tarjeta de nodo es el hub. El hub tiene una conexión directa llamada SysAD bus, que lo comunica con la memoria principal de la tarjeta de nodo. Esta conexión provee un ancho de banda neto de 780 MB/s, el cual es compartido por los dos R10000 de la tarjeta.

El acceso a la memoria remota (memoria de otra tarjeta de nodo) se hace mediante una conexión llamada CrayLink. Ésta es una conexión bidireccional que une al ruteador con el hub. Su ancho de banda neto es de 780MB/s en cada dirección. El ancho de banda efectivo para el usuario es de 600 MB/s en cada dirección. El decremento en el ancho de banda se debe a la transmisión de información de control, aproximadamente 32 bytes por paquete transmitido [SIL96].

El hub también controla el acceso de los dispositivos de I/O (entrada/salida) a través de un canal llamado XIO.

Después de ver de forma general todas las características de los nodos de ORIGIN 2000, podríamos preguntarnos, ¿ Qué sucede si se tiene ejecutando un programa en dos nodos diferentes y cada uno de los nodos desea alterar alguno de los datos, se actualiza el dato modificado en la otra copia del programa?, a continuación se tratará de responder esta pregunta. Cada tarjeta de nodo de ORIGIN tiene su memoria privada, por lo que puede correr independientemente cualquier programa, no es raro que un programa se ejecute en varios nodos, ya que cada nodo puede tener su propia copia del programa. Supongamos que un nodo está ejecutando parte de un programa, al mismo tiempo otro nodo del sistema está corriendo otra parte del programa, los dos están compartiendo los mismo datos. Si el primer nodo trata de alterar un dato de alguna estructura de control (por ejemplo algún driver del kernel), escribirá el nuevo valor en su memoria local, por lo que existirán dos valores para el mismo dato. El problema anterior se resuelve mediante el sistema de coherencia de caché. Este mecanismo utiliza la información guardada en el *directorio* y checa una serie de bits para ver si se tienen permisos de modificación sobre esa área de memoria, en caso de tenerlos modifica los datos almacenados en esa porción de la memoria. Cada palabra almacenada en la memoria del nodo tiene una cadena asociada de permisos, los cuales se guardan en el *directorio* del nodo. Esta cadena se actualiza de forma global en cada modificación de los datos. Si un proceso no tiene acceso a la memoria, ejecuta otras instrucciones hasta que se libera la memoria. Si por el contrario tiene permisos, envía actualizaciones de la línea de directorio de ese dato a los otros procesadores, con lo que se previenen modificaciones de los datos por otros procesadores. Una vez terminada la operación, el procesador vuelve a enviar la información del directorio a los demás nodos, notificando que el dato ya fue liberado y su nuevo valor. En este momento, cualquier nodo puede alterar el valor del dato en cuestión [SIL96].

La conexión XIO nos permite comunicarnos con un chip controlador de dispositivos, el XBOW. Este chip puede conectar tarjetas ethernet, inerfaces FDDI, controladores SCSI, CDROM's y discos duros.

Finalmente hablaremos de las configuraciones que se pueden construir con nodos de ORIGIN 2000. Los sistemas de 2 y 4 procesadores, consisten únicamente de dos tarjetas de nodo conectadas directamente mediante el hub. En el caso de sistemas de 8 procesadores, estos consisten de 2 ruteadores unidos mediante una conexión CrayLink. Cada ruteador tiene conectado 2 nodos. Además de ésto existen conexiones a 2 chips XBOW entre los nodos de cada ruteador. Esta conexión se muestra en el diagrama izquierdo de la figura 7.11. Esta conexión es conocida como 8P12.

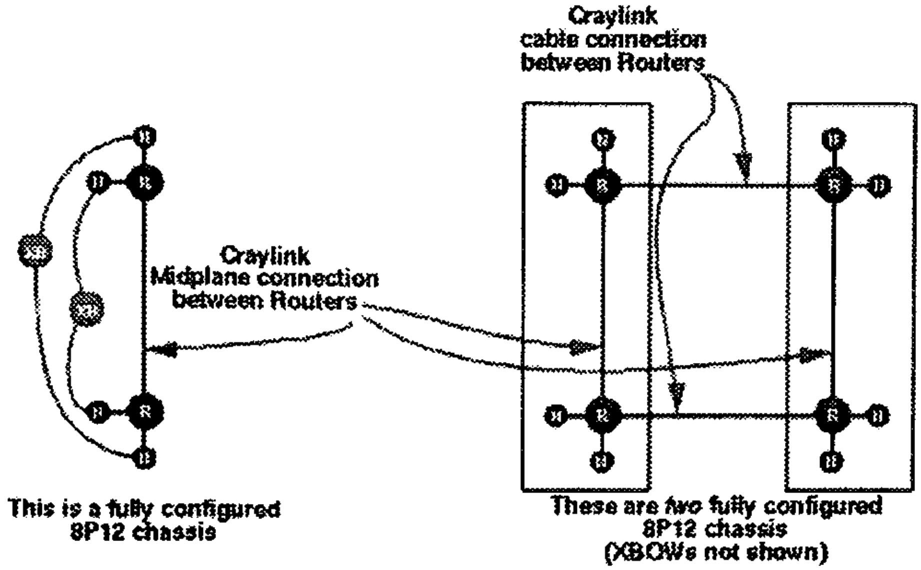


Figura 7.11. Conexión 8P12.

El diagrama de la derecha de la figura 7.11 muestra un sistema de 16 procesadores. En el caso de un sistema de 32 procesadores, es necesario unir 4 módulos 8P12 como se muestra en la figura 7.12.

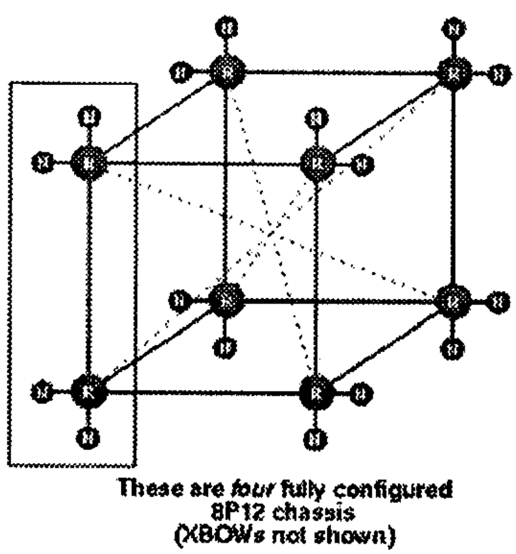


Figura 7.12. Sistema de 32 procesadores.

La figura 7.12 nos muestra una conexión de hipercubo de dimensión 3. Como se mencionó anteriormente, cada ruteador tiene 6 puertos, por lo que en esta conexión todavía tenemos un puerto disponible en cada ruteador, el cual podremos utilizar para unir cada uno de sus vértices con los vértices de otro hipercubo de dimensión 3, con lo que construiríamos un hipercubo de dimensión 4, es decir, una máquina de 64 procesadores. En la figura 7.13 se muestra un sistema de 64 procesadores.

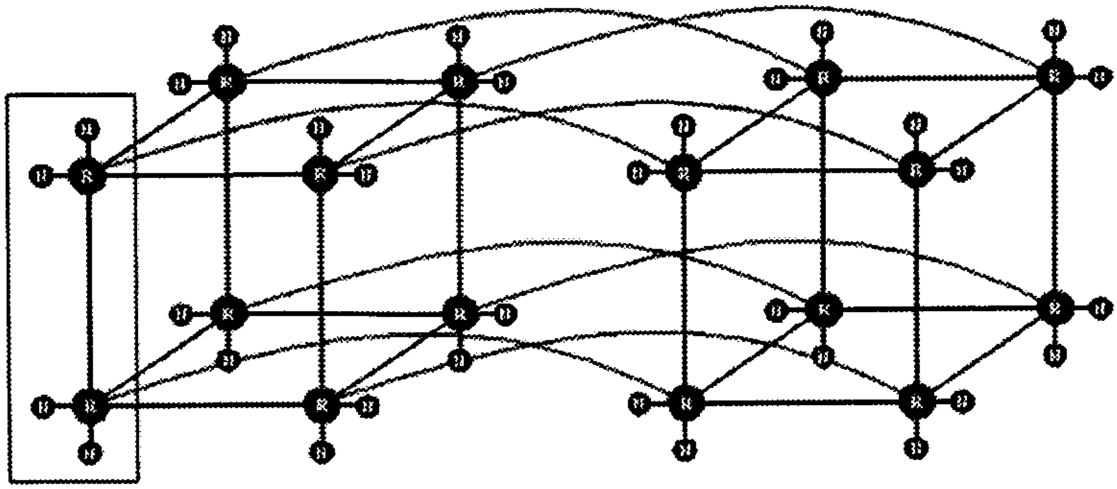


Figura 7.13 . Sistema de 64 procesadores.

En la conexión mostrada en la figura 7.13, ya no hay puertos disponibles en los ruteadores. Por esta razón para los sistemas de 128 procesadores, debe de comenzarse a construir 4 sistemas de 32 procesadores. Como vimos anteriormente, en este tipo de sistemas existe un puerto (conexión CrayLink) libre por cada ruteador. En lugar de conectar directamente las líneas libres de los nodos, éstas deben de conectarse a un ruteador. En cada ruteador convergirán los nodos correspondientes de los 4 hipercubos de dimensión 3. La figura 7.14 muestra esta conexión.

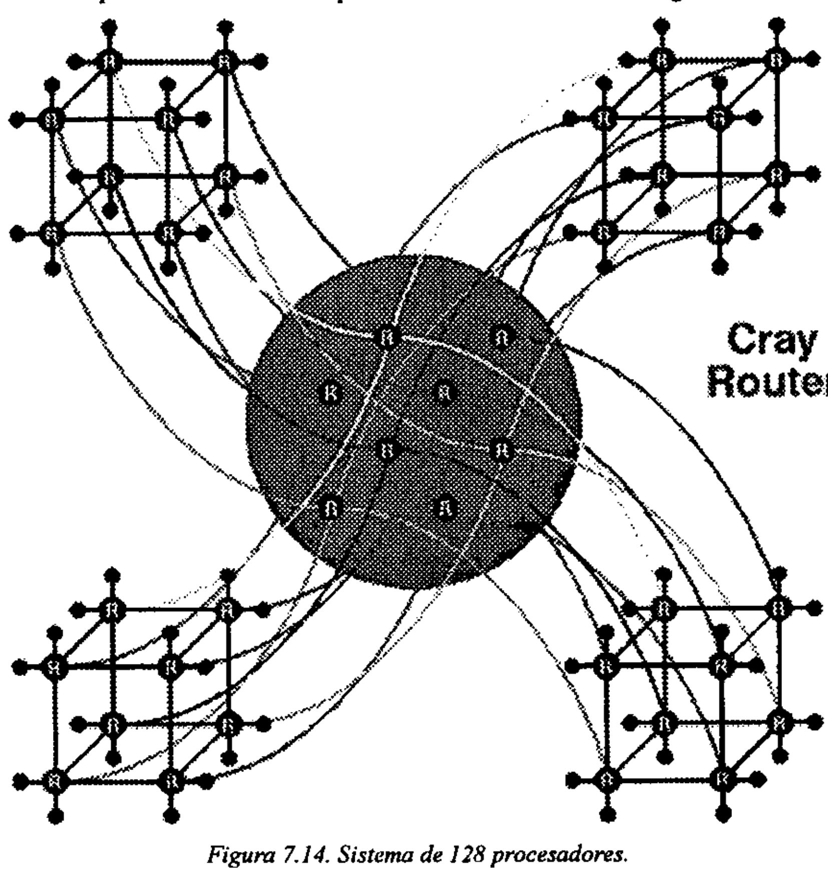


Figura 7.14. Sistema de 128 procesadores.

En la figura 7.14, podemos ver que para esta conexión se necesitan ocho ruteadores adicionales para unir a los 4 sistemas de 32 procesadores. Estos ruteadores vienen en un gabinete separado, el cual se conoce como Cray Router. Sistemas de más de 128 procesadores sólo se pueden conectar bajo el esquema de arreglo (cluster) [SIL96].

Finalmente cabe mencionar que la supercomputadora CRAY-ORIGIN 2000 de la UNAM está dividida en 2 máquinas, una de 8 y otra de 32 procesadores, ésto debido a políticas de uso. Cuenta con 170 GB en disco duro y 10 GB de memoria principal.

7.3 Pruebas de ejecución del espacio virtual.

Las pruebas de ejecución son las mismas que en el caso de la O2, es decir, consisten de un recorrido en el espacio virtual utilizando las herramientas de software propuestas. La única diferencia es que en este caso se obtuvieron más datos acerca de la corrida de las herramientas. Ésto se pudo hacer debido a que los procesadores R10000 tienen contadores internos los cuales soportan el monitoreo de varios eventos, entre los cuales se encuentra el ancho de banda de memoria, la cantidad de MegaFLOPS y el tiempo de corrida.

7.4. Observaciones y resultados.

En las máquinas paralelas existen muchos parámetros por medio de los cuales se puede medir el desempeño de un programa. Los procesadores R10000 auxilian en esta tarea, ya que cuentan con contadores de eventos implementados en hardware como lo muestra la figura 7.15.

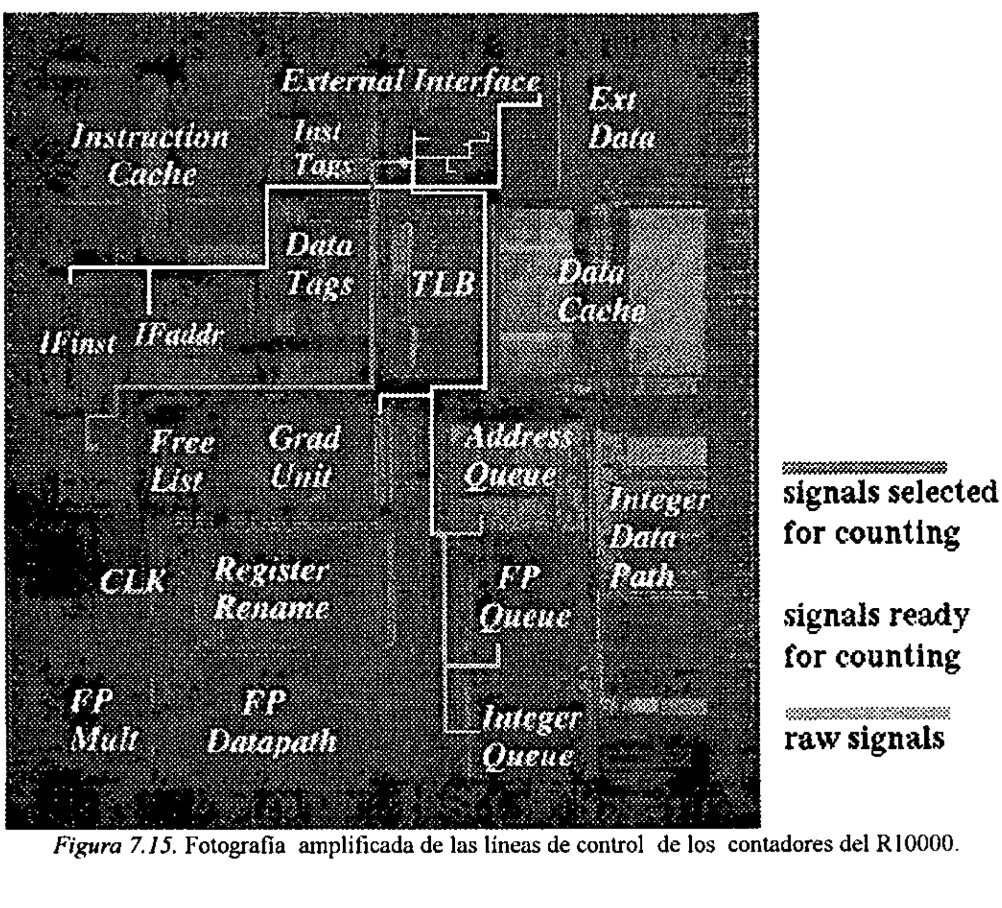


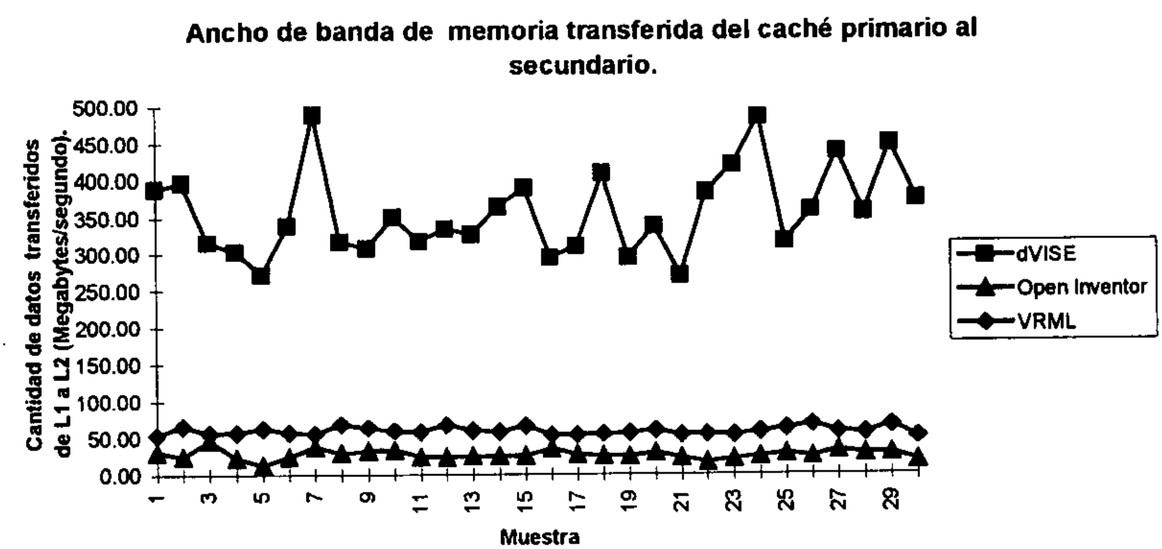
Figura 7.15. Fotografía amplificada de las líneas de control de los contadores del R10000.

Estos contadores pueden proporcionar información acerca de 32 eventos. La información de estos contadores se puede obtener mediante el comando *perfex(1)*.

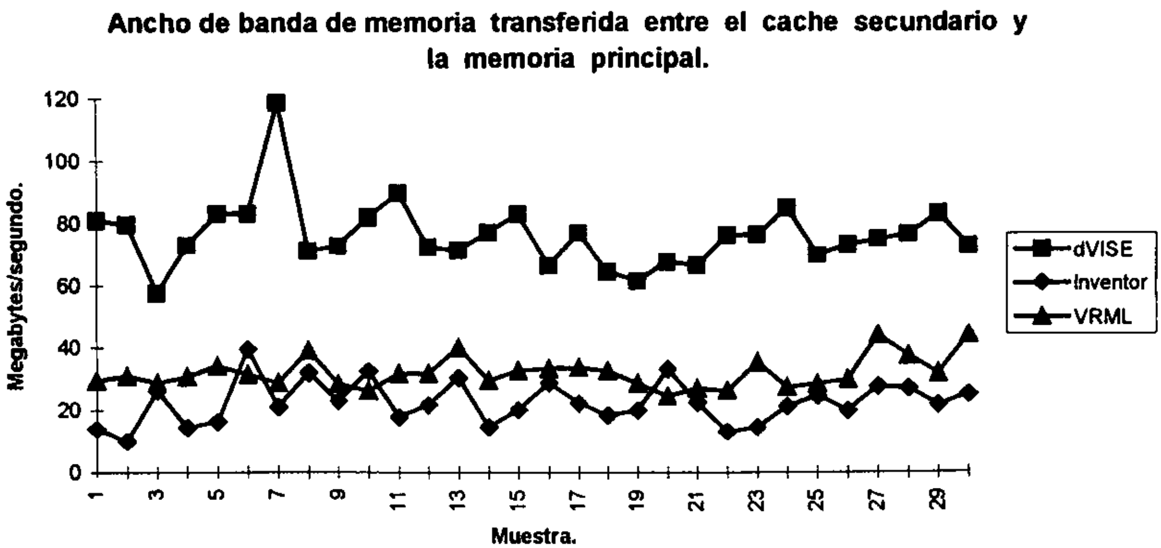
Para los objetivos de este trabajo sólo se tomaron en cuenta los siguientes eventos:

- Ancho de banda de memoria transferida del caché primario al caché secundario.
- Ancho de banda de memoria transferida del caché secundario a la memoria principal.
- Número de instrucciones de punto flotante realizadas por segundo (MFLOPS)
- Tiempo real, tiempo de usuario y tiempo de sistema.

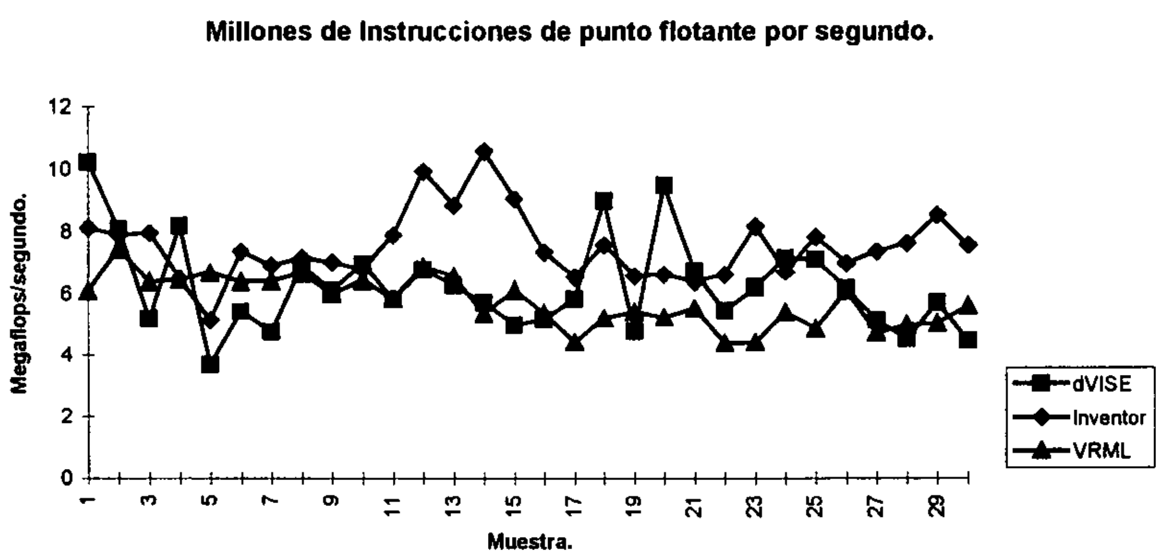
A continuación se muestran las gráficas de comportamiento. Gráfica 7.1,7.2,7.3,7.4,7.5 y 7.6:



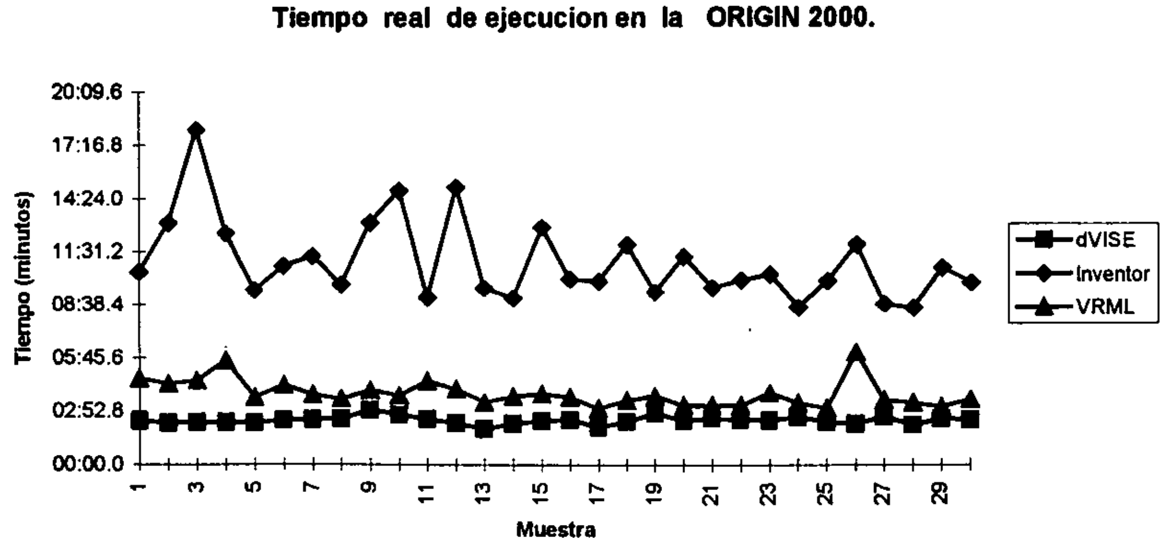
Gráfica 7.1. Ancho de banda transferido del caché primario al secundario.



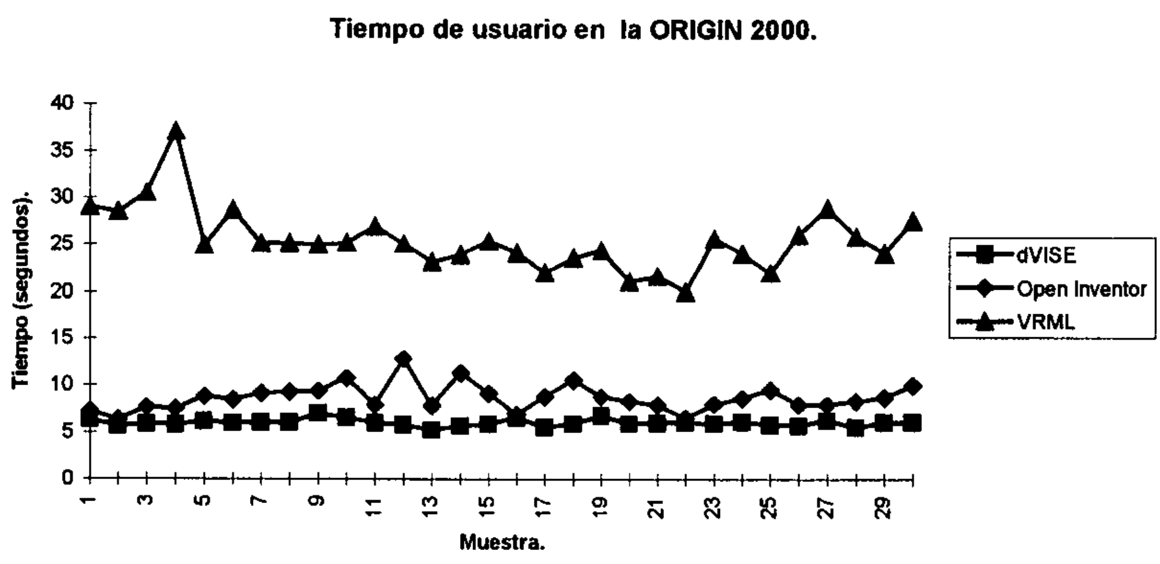
Gráfica 7.2. Ancho de banda transferido del caché secundario a la memoria principal.



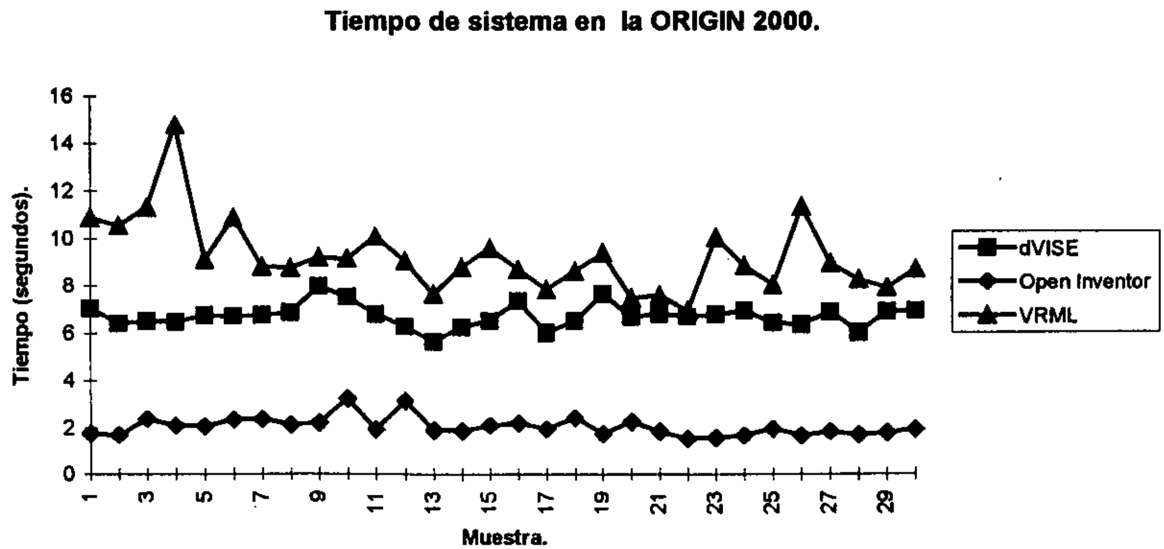
Gráfica 7.3. Millones de Instrucciones de punto flotante por segundo.



Gráfica 7.4. Tiempo real de ejecución en la ORIGIN 2000.



Gráfica 7.5. Tiempo de usuario en la ORIGIN 2000.



Gráfica 7.6. Tiempo de sistema en la ORIGIN 2000.

Los resultados promedio fueron:

Parámetro	dVISE	Open Inventor	VRML
Ancho de banda de memoria transferida entre el caché primario y el caché secundario (MBytes/segundo).	356.09	25.60	58.14
Ancho de banda de memoria transferida entre el caché secundario y la memoria principal (MBytes/segundo)	76.17	22.39	31.96
Mega FLOPS	6.25	7.50	5.74
Tiempo real (minutos)	02:20.8	10:57.0	03:47.1
Tiempo de usuario (segundos)	5.99	8.68	25.50
Tiempo de sistema (segundos)	6.71	2.02	9.23

Para la ORIGIN 2000, un programa tienen un buen desempeño, si consume poco tiempo de procesamiento en CPU (tiempo de usuario) y realiza el mayor número de operaciones de punto flotante posible (MFLOPS) [DAN97].

El tiempo real nos indica que tan fácil o difícil es la navegación en un producto, por ejemplo, en el caso de Open Inventor se ve claramente que su interfaz no es adecuada para navegar. Lo mismo sucede con VRML (Cosmo Player), ya que aunque su interfaz es un poco mejor que la de Inventor, sigue presentando problemas al navegar, aunado a ésto tenemos que el visor de VRML es una capa de otro producto, por lo su desempeño se ve condicionado al comportamiento del navegador (browser).

dVISE en términos generales es la herramienta que mejor desempeño tuvo en la ORIGIN 2000. Sin embargo es la aplicación que requiere más memoria, ya que utiliza ancho de banda grande, tanto en transferencia de datos entre cachés, como en el intercambio de información con la memoria principal.

En el caso del tiempo de sistema, vemos como Open Inventor es la herramienta que hace menos llamadas al sistema.

En cuanto a los MegaFLOPS, podemos decir que las tres aplicaciones no tienen un buen desempeño, ya que ni siquiera llegan a los 10 MegaFLOPS. Para que pueda decirse que un programa está utilizando de forma eficiente los recursos de una supercomputadora como la ORIGIN, debe de alcanzar los 200 MegaFLOPS como mínimo, por lo que las tres herramientas subutilizan los recursos de cómputo de la CRAY-ORIGIN 2000.

Finalmente cabe mencionar que todas las herramientas son susceptibles de mejorar su rendimiento. Ésto sería posible si se contara con el código fuente de cada una de ellas, sin embargo, como son productos comerciales no es posible tener acceso a esta información, por lo cual tendremos que esperar hasta que los fabricantes mejoren sus productos para poder aprovechar las ventajas que nos ofrece una máquina como la ORIGIN 2000.

Capítulo 8.

Conclusiones y análisis de resultados.

De acuerdo con los resultados presentados en el capítulo 6 y en el capítulo 7, la herramienta que mejor desempeño tuvo fue dVISE, ya que además de presentar los tiempos más bajos de todas las aplicaciones, registró un buen número de MegaFLOPS. En segundo lugar tenemos a VRML con su visor Cosmo Player, el cual registró buenos tiempos y una cantidad aceptable de Mega FLOPS. En último lugar tenemos a Open Inventor con su visor *ivview* utilizando programación ASCII.

La causa de que VRML e Inventor tuvieran desempeños pobres fue su forma de navegar y manipular los objetos del espacio virtual. Ésto se ve claramente reflejado en los tiempos "reales" de la aplicación, en el ancho de banda de memoria y en el tiempo de "sistema". De las gráficas del ancho de banda de memoria, podemos concluir que tanto VRML como Inventor hacen mucho acceso a disco duro, debido a que sus anchos de banda son muy pequeños, por lo que, los datos de la geometría son llevados a memoria cuando éstos aparecen en la escena actual, lo cual trae como resultado un incremento en las llamadas de sistema, con lo que aumenta el tiempo de sistema; aunque en el caso de Inventor en la ORIGIN, podemos apreciar como su tiempo de sistema es pequeño, lo cual nos indica que todas las llamadas al sistema se hacen para abrir y cerrar archivos.

Como se mencionó en el capítulo 7, el factor de *speed-up* es un indicador de que tan buen comportamiento tiene un programa en una plataforma paralela. Sin embargo no fue posible obtenerlo debido a que la estación de trabajo O2 tiene un procesador R5000, mientras que la ORIGIN 2000 tiene procesadores R10000, por lo que no es posible obtener este factor. Para solucionar este problema, se trato de correr todas las herramientas en un sólo procesador de la ORIGIN 2000. Ésto no fue posible, debido a que para restringir el uso de los procesadores de la ORIGIN, es necesario correr un comando que sólo el administrador del sistema tiene permiso de usar. Se habló con el administrador de la ORIGIN 2000, pero no fue posible modificar el número de procesadores asignados a la ejecución de los programas del sistema, ya que si se modificaba esta configuración, los demás usuarios de la máquina se verían afectados.

Por lo cual, para poder tener algún parámetro de comparación, se invitó a que varios usuarios hicieran pruebas de las aplicaciones corriendo en las dos máquinas. Todos ellos dijeron que las aplicaciones corrían más rápido cuando se ejecutaban desde la ORIGIN 2000 y que la herramienta que permitía una navegación más fácil era dVISE. Cabe señalar que en el caso de la ORIGIN, se utilizó una de las máquinas del Laboratorio de Visualización, las cuales están conectadas mediante un anillo de FDDI[1] a la ORIGIN 2000, por lo que la velocidad de la red no alteró el desempeño de las aplicaciones.

[1] Fibra Óptica.

Cuando nosotros tenemos un programa y lo corremos en una máquina paralela esperamos que el tiempo de ejecución disminuya en una proporción considerable, sin embargo, ésto muchas veces no es posible debido a que los algoritmos del programa tal vez no sean óptimos, o quizá el código sea en su totalidad secuencial, por lo que muy probablemente el programa corra más rápido en una computadora de un sólo procesador. La causa de ésto, es que un código secuencial necesariamente tiene que esperar a que la instrucción anterior se haya ejecutado para poder realizar la siguiente, por lo que, cuando nosotros migramos a una máquina paralela un programa secuencial, el código se distribuye a varios procesadores, con lo que en lugar de ayudarse se estorban unos con otros, recordemos que la ORIGIN tiene un sistema de coherencia de caché, el cual verifica que ningún dato sea alterado de forma indebida por el sistema. Esta validación de datos consume mucho tiempo, por lo que el desempeño del programa se va a pique. Por esta razón debemos de tener bien claro que tipo de programa y de código deseamos correr en forma paralela. Los programas que tienen un aprovechamiento óptimo son los de métodos numéricos, ya que tienen muchos ciclos iterativos, que son fáciles de paralelizar. Por otra parte, los programas con mucho código secuencial como los sistemas expertos, tienen un desempeño bajo y difícilmente llegan a mejor su tiempo de sistema, de usuario y real.

Pueden existir códigos secuenciales cuya demanda de memoria sea demasiado grande como para correr en una estación de trabajo, en este caso el desempeño se debe de analizar por la cantidad de memoria accesada por el programa durante un determinado período de tiempo.

Como podemos ver, el desempeño de una aplicación pude medirse en base a varios parámetros, sin embargo casi siempre se hace con respecto al tiempo, debido a que es el recurso no renovable más valioso para todas las personas.

Otro factor que puede decrementar el desempeño de un programa es la forma en que se hayan compilado las bibliotecas utilizadas por él. El R5000 y el R10000 son procesadores de 64 bits. Los programas (y las bibliotecas) compiladas en sistemas basados en ellos, tienen la opción de compilarse a 32 bits o a 64 bits. Haciendo un breve paréntesis, bajo el sistema operativo UNIX existen dos tipos de programas, los dinámicamente ligados y los estáticamente ligados. Un programa estáticamente ligado incluye todas las bibliotecas utilizadas en tiempo de compilación, es decir, las hace parte del código ejecutable. Mientras que los dinámicamente ligados cargan las bibliotecas al tiempo de la corrida. Los programas utilizados en plataforma Silicon Graphics, en su mayoría son dinámicamente ligados, por lo que su tamaño es pequeño. El ligado dinámico puede ser un problema cuando migramos nuestro programa a una plataforma superior. En el caso de dVISE, las bibliotecas que proporciona DIVISION están compiladas en 32 bits, por lo que no aprovechan al máximo las capacidades de la ORIGIN.

Con respecto a la red, cabe mencionar que el nodo de red que se tiene en el Laboratorio de Interfaces Inteligentes es muy lento, ésto se puede apreciar en la gráficas del capítulo 5. Las aplicaciones gráficas cuando corren remotamente, hacen una petición al servidor de despliegue "X[1] " de la máquina local, para que interprete la información que se le envía y la despliegue en pantalla. En el caso de las herramientas de realidad virtual, el cliente (en este caso podría ser la ORIGIN) envía la información de la geometría de la escena que quiere desplegar, por medio de paquetes de datos dirigidos al servidor, el servidor toma esta información y valiéndose de las bibliotecas gráficas locales despliega la escena en pantalla. El tamaño de los paquetes varia, en el peor de los casos el tamaño podría ser de 12.5 Mbytes. Por lo que requeriríamos esa información cada segundo. Ésto nos da un ancho de banda de 12.5 Mbytes por segundo o 100 Mbits por segundo. Por lo que, para tener un buen desempeño en la utilización de espacios virtuales remotos, es necesario tener un enlace de red que soporte esta velocidad. En el caso del Laboratorio de Interfaces Inteligentes, una solución a este problema podría ser una conexión directa de la máquina O2 del laboratorio a la supercomputadora CRAY-ORIGIN 2000, o en su defecto una conexión directa al backbone de redUNAM.

[1] X ó X11 es un estándar de ventanas y de aplicaciones gráficas que se utiliza bajo ambiente UNIX.

A continuación mencionaremos las ventajas y desventajas de cada una de las herramientas empleadas durante este trabajo:

- *dVISE*. Una de las principales ventajas de dVISE es la facilidad con la cual se pueden crear espacios virtuales utilizando herramientas de CAD accesibles para cualquier usuario. Por otra parte soporta varios dispositivos de realidad virtual, entre los cuales encontramos a los cascos (Head-Mounted Displays), los guantes de datos y el Mouse 3-D. Se pueden agregar comportamientos físicos a los objetos mediante varias herramientas proporcionadas en menús tridimensionales. Soporta mapeo de imágenes para la creación de espacios virtuales más detallados, además de permitir agregar programas del usuario escritos en lenguaje C, con lo que se aumenta el poder de la aplicación. Su mayor desventaja es su costo.

- *VRML* . Su principal ventaja es que puede ser utilizado en cualquier plataforma donde corra un navegador de Internet, sin embargo algunos visores no pueden ser utilizados en todas las plataformas, por ejemplo Cosmo Player no corre en máquinas SUN. Para agregar comportamientos a los objetos es necesario programar en Java. Su sistema de navegación es bueno y se tiene un control completo de los eventos y de las rutinas utilizadas en el espacio. Una gran desventaja es que sólo soporta la comunicación mediante el Mouse, por lo que en caso de requerir la utilización de otro dispositivo de navegación se tendría que programar la interfaz. Su mayor ventaja es que es gratuito y es muy fácil de programar, debido a que se programa por medio de directivas, es decir, en ASCII. Por otra parte existen muchos convertidores de formato en Internet que permiten que cualquier geometría hecha en otros paquetes sea exportada a VRML, dándole versatilidad al lenguaje .

- Open Inventor. Su visualizador *ivview* no está diseñado para navegar, sin embargo es soportado por todas las plataformas de hardware y software de Silicon Graphics. Se puede programar espacios virtuales utilizando C y C++, además de la programación en ASCII, con lo que se tiene el control completo del programa. Sin embargo, cuando se programa en C o C++ es necesario definir toda la geometría, dando las coordenadas de cada vértice del espacio virtual, ya que sólo se pueden utilizar algunas primitivas (cilindro, cubo, etc.), lo cual limita y retarda el diseño de espacios virtuales. Su programación en ASCII es compleja y el tamaño de los archivos de geometría que genera es grande, por lo que se recomienda el uso de programas de CAD para el modelado de objetos (o espacios) y convertidores de formato de geometrías para poder importar nuestro modelo a Open Inventor. Finalmente cabe mencionar que la mayor ventaja de Open Inventor es la utilización de todos los recursos gráficos de la máquina y la amplia variedad de funciones de biblioteca que lo componen.

Por el lado del hardware, podemos observar como la O2 registro tiempos mas pequeños que la ORIGIN 2000, ésto se debió a que la arquitectura de la supercomputadora no está orientada al procesamiento de gráficos, sino al procesamiento numérico, por lo que lo ideal sería correr el software de realidad virtual en una máquina Onyx2, debido a que este tipo de máquinas tienen una arquitectura igual a la de la ORIGIN 2000, pero orientada al procesamiento de gráficos. Aún así, la ORIGIN 2000 registró tiempos muy buenos, lo cual nos muestra la velocidad de procesamiento que puede alcanzar esta máquina.

Como comantario adicional se mencionaran las características más importantes de la ORIGIN 2000 y de la O2:

- Supercomputadora CRAY-ORIGIN 2000. Se compone de tres gabinetes divididos en dos módulos cada uno. Cada módulo cuenta con 8 procesadores. Cada gabinete mide 1.85 m de alto, 71 cm de ancho, y 1.02 m de fondo. Su peso es de 1,200 Kg., en total se consumen 6.2 KW/hr. La capacidad de los discos de la Origin2000 es de 170 GB. Son unidades SCSI de 9.1 GB cada una, que están dentro de los gabinetes. Los medios de comunicación a través de la red son interfaces Ethernet, FDDI y ATM.Con un rendimiento teórico pico de 15.60 Gflops [DEP98].

- Estación de trabajo O2.Tiene un procesador R5000 de 180 Mhz..64 MB en RAM. Mide 30 cm. de alto, 22.5 cm. de largo y 12.5 de ancho. Cuenta con 4 GB en disco duro. Tiene una interfaz de red 100 BaseT. Cuenta con una cámara de video, salidas de audio y video digital.

Otro aspecto importante de señalar, es el hecho de que ninguna de las herramientas de software de realidad virtual probadas cuenta con bibliotecas de 64 bits, ésto se debe posiblemente a que los fabricantes están esperando agregar nuevas capacidades a sus productos y a corregir algunos errores, por lo es muy probable que las bibliotecas de 64 bits aparezcan con las nuevas versiones de cada una de las herramientas, con lo que se espera tener un mejor desempeño de las mismas.

Uno de los objetivos de este trabajo fue instalar diferentes herramientas de software de realidad virtual en la supercomputdora CRAY-ORIGIN 2000, además de analizar su comportamiento. Las herramientas no tuvieron ningún problema para ser instaladas en la supercomputadora, ni tampoco al momento de ser corridas. Por lo que podemos asegurar que no existirá ningún problema de compatibilidad con equipo Silicon Graphics.

El uso del supercómputo en la realidad virtual no es muy común, debido a que el tiempo de procesamiento es muy costoso. En la U.N.A.M. estámos siendo pioneros en el uso de supercomputadoras Silicon Graphics para la creación de espacios virtuales y en sí de la telepresencia.

Finalmente cabe mencionar que este trabajo mostró las ventajas y desventajas de correr software de realidad virtual bajo ambientes paralelos y bajo ambientes de un sólo procesador. Por otra parte se mostró que es necesario optimizar al máximo los recursos de cómputo de los que disponemos, ya que cada vez nos acercamos más al límite de velocidad de los microprocesadores, por lo que, para elevar el desempeño de los programas y aumentar la velocidad de procesamiento las alternativas que tenemos son: el trabajo paralelo y la optimización de recursos, algoritmos y código de programa. En ocasiones hemos visto como al inicio de la creación de un sistema nos preocupamos únicamente por obtener los requerimientos del proyecto y nunca nos ocupamos de analizar que tan eficientemente está utilizando mis recursos de cómputo el compilador o la herramienta de desarrollo que estoy empleando. Esta costumbre debe de cambiar, ya que es labor del ingeniero en computación obtener toda esta información y decidir cuál es el mejor camino a seguir, es decir, elegir la herramienta que más se adecue a los requerimientos del proyecto y que empleé de forma óptima los recursos de cómputo de los que se dispone, con ésto lograremos tener software rápido, pequeño y que aproveche de forma óptima la plataforma de hardware en la que lo estámos corriendo.

Anexo A

Ilustraciones.

Anexo B

Gráficas.

Glosario.

ASCII.- American Standar Code for Information Interchange, es un código que traduce carácteres alfabéticos, numéricos así como símbolos e instrucciones de control en un código binario de siete u ocho bits.

Backbone.- Red de alta velocidad que permite comunicar varias redes de computadoras, generalmente redes de área local.

Bibliotecas.-Conjunto de funciones compiladas que pueden ser accesadas por un programa. Estas funciones están en código objeto, por lo que pueden ser agregadas al programa al momento de ligarlo.

Browser.-Navegador o visualizador. Es un programa que permite navegar o buscar información en Internet, mediante documentos de hipertexto.

Memoria compartida.- Esquema de trabajo de cómputo paralelo, en el cual la memoria es común para todos los procesadores.

Memoria distribuida.- Esquema de trabajo de cómputo paralelo en el que cada procesador tienen su propia memoria y comparte datos e instrucciones mediante el intercambio de mensajes.

Overhead.- Información que se agrega a los paquetes de datos transmitidos entre procesadores para sincronizar la comunicación.

Proceso.- Un proceso es un programa en ejecución que consta del programa ejecutable, sus datos, pila y del valor de los registros asociados a la ejecución del mismo.

Speed-up.-Factor que nos indica en que proporción se ha modificado el desempeño de un programa en una máquina paralela.

Throuhgput.- Número de trabajos producidos por un sistema en un intervalo de tiempo.

Sección crítica.- Modificación o acceso a algún recurso de cómputo compartido.

Bibliografía

1. Libros.

[AND96] Anderson, Pat. ***Alias Overview***. Ed. Silicon Graphics, 1996 U.S.A..

[DAN97] Dankwardt, Kevin. ***IRIX Parallel Programming. Student handbook.*** Ed. Silicon Graphics, 1997 U.S.A..

[DEZ97] Dezotti Ruiz Alexei, Figueroa Calix Miguel, Gonzalez Guizar Alberto, Santos Jaimes German, Vazquez Gonzalez Luis. ***Construcción del espacio virtual de la remodelación de la planta baja del instituto de geografía.*** Facultad de Ingeniría U.N.A.M. , 1997 México.

[DIV297] Division Co. ***DV/Reality for UNIX Workstations User Guide.*** Ed. Division, 1997 U.S.A. .

[DIV397] Division Co. ***dVISE for UNIX Workstations Geometry Tools.*** Ed. Division, 1997 U.S.A. .

[DIV97] Division Co. ***dVISE for UNIX Workstations Configuration Guide.*** Ed. Division, 1997 U.S.A. .

[FAC97] Facultad de Ciencias. ***Memorias del congreso Computación Visual 97.*** Ed. U.N.A.M., 1997 México.

[GOL93] Golup G., Ortega James M.. ***Scientific computing: An introducction to parallel computing.*** Ed. Academic Press, 1993 U.S.A..

[GRA97] Gradecki, Joe. ***Realidad virtual.*** *Construcción de proyectos.* Ed. Ra-ma, 1997 México.

[HAM93] Hamit, Francis. ***Virtual Reality and the exploration of cyberspace.*** Ed. Sams publishing, 1993 U.S.A..

[HAR96] Harrison, David. Jaques, Mark. ***Experimentos de Realidad Virtual.*** Ed. Butterworth Heimemann, 1996 U.K. .

[IMM95] Immier, Christian. ***El gran libro del autodesk 3D studio Versión 4.0***. Ed. Marcombo, 1995 España.

[MAR96] Marquez Garcia, Francisco Manuel. ***Unix programacion avanzada.*** Ed. Ra-Ma, 1996 España.

[MOR95] Morrison, Mike. ***The magic of computer graphics.*** Ed. Sams publishing, 1995 U.S.A.

[PRES93] Pressman, Roger S.. ***Ingenería del software, un enfoque práctico.*** Ed. Mc Graw Hill, 1993 Méxio.

[TAN93] Tanenbaum, Andrew S.. ***Sistemas operativos modernos.*** Ed. Prentice Hall, 1993 México.

[VIN95] Vince, John. ***Virtual reality systems.*** Ed. ACM Press, 1995 U.S.A.

[WER94] Wernecke, Josie. ***The inventor Mentor.*** Ed. Addison Wesley, 1994 U.S.A.

[WIL95] Wilson, Rodney C.. ***UNIX test tools and benchmarks.*** Ed. Prentice Hall, 1995 U.S.A.

2.Sitios en Internet.

[DEP98] Departamento de Supercómputo,DGSCA, UNAM. ***Semana del supercómputo en Méxco***.UNAM 1998.
http://www.super.unam.mx/O2000.html .

[MAR96] Marco Zagha, Brond Larson, Steve Turner, Marty Itzkowitz. ***Performance Analysis Using the MIPS R10000 Performance Counters.*** Proceedings Supercomputing'96, November 1996, Pittsburgh, PA Copyright © 1996, IEEE Computer Society.
http://www.sgi.com/MIPS/products/r10k/App_notes/Perf_Count/index.html

[RUD97] Rudomin, Issac. ***Curso de VRML***. ITESM 1997 Mexico.
http://sgio2.cem.utesm.mx/rudomin/VISUALPS/pressbuton.hmtl

[SIL96] Silicon graphics. ***Performance Tuning Optimization for Origin2000 and Onyx2***. Silicon Graphics, 1996 U.S.A.
http://techpubs.sgi.com/library/dynaweb_bin/0640/bin/nph-dynaweb.cgi/dynaweb/SGI_Developer/OptOrig/@Generic__BookView/36

[SIL296] Silicon Graphics. ***Open Inventor***. Silicon Graphics 1996 U.S.A..
http://www.sgi.com/Technology/Inventor.html.

[SIL396] Silicon Graphics.***VRML, Cosmo Player***. Silicon Graphics 1996 U.S.A..
http://vrml.sgi.com

[SIL496] Silicon Graphics.***O2 Cracteristics***.Silicon Graphics 1996 U.S.A..
http://www.sgi.com/Products/hardware/desktop/products/arch.html#O2

Resultados y Conclusiones.

Las pruebas mostraron que dVISE es el producto que tiene mejor desempeño, seguido por VRML y finalmente por Open Inventor.

Aunque parezca contradictorio, dVISE está basado en las bibliotecas de Open Inventor. Sin embargo debemos recordar que lo que se probó en este trabajo fue la programación en modo ASCII de Open Inventor, no la programación con la biblioteca de funciones en C++ ó C. La programación ASCII es interpretada, por lo que la ejecución es más lenta.

dVISE está programado en C, por lo que su ejecución es más rápida, notándose claramente esta diferencia en las pruebas realizadas durante este trabajo.

El código de VRML también es interpretado, por lo que es lento en su ejecución.

En cuanto a memoria, dVISE es la herramienta que más memoria consume, debido a que casi no emplea acceso a disco, mientras que las otras herramientas lo accesan frecuentemente, ya que tienen que leer el código del espacio virtual para poder actualizar las vistas del mismo.

Finalmente cabe señalar que existen algunas diferencias entre los tiempos de ejecución del software en la O2 y en la CRAY-ORIGIN 2000, ésto se debe a que las herramientas están implementadas con mucho código secuencial y poco código paralelizable, por lo que el desempeño y el tiempo de ejecución no varia mucho al utilizar la supercomputadora.